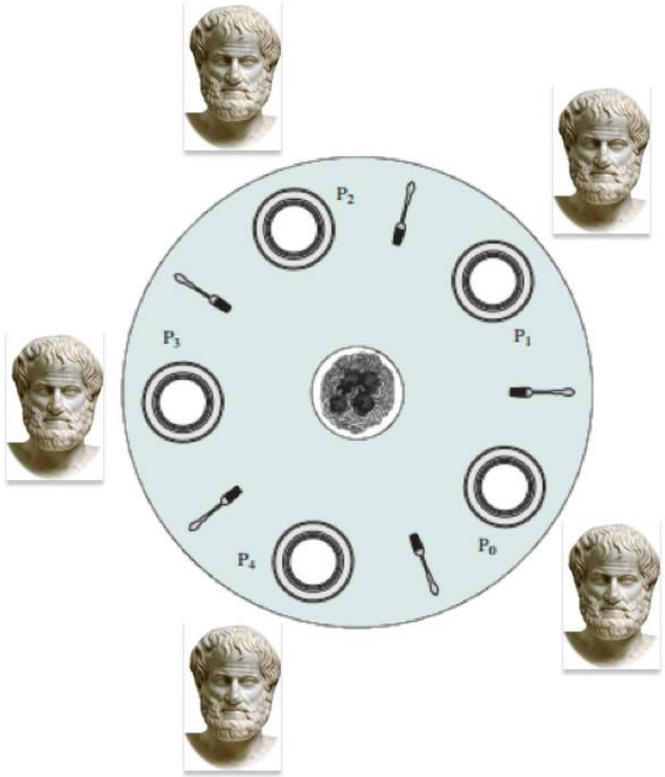
Operating Systems:
DINING PHILOSOPHERS

Nguyễn Tống Minh - 20204885
Nguyễn Công Đạt - 20200137
Ngô Thị Thu Huyền - 20200289
Nguyễn Trung Hiếu - 20204877

INTRODUCE
THE PROBLEM

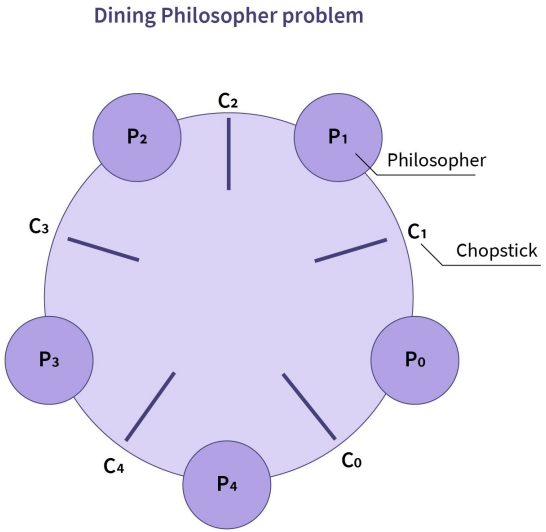


Dining Philosophers Problem

Consider two processes P1 and P2 executing simultaneously, while trying to access the same resource R1

Who will get the resource and when?

- Assume there are K philosophers sat around a circular table
- Each with one fork between them
- A philosopher can eat only if he/she can pick up both the forks next to him/her
- One of the adjacent followers may take up one of the forks, but not both.

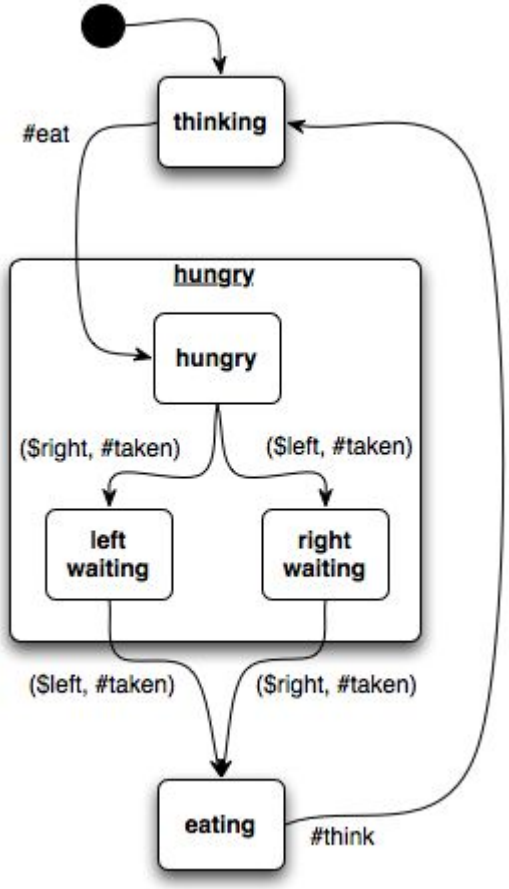


Real Situations

- think until the left fork is available; when it is, pick it up
- think until the right fork is available; when it is, pick it up
- when both forks are held, eat for a fixed amount of time
- put the left fork down
- put the right fork down
- repeat from the beginning

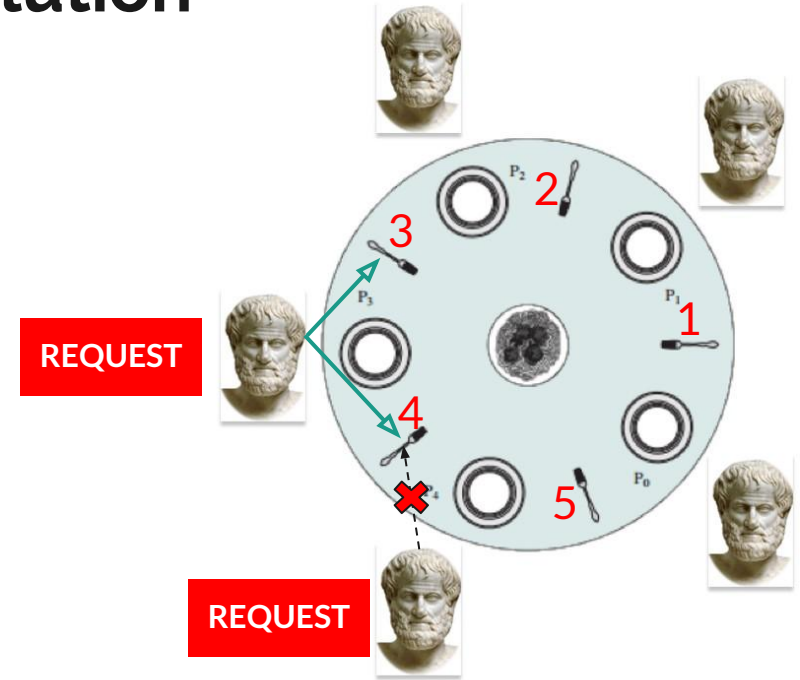
DEADLOCK

METHODOLOGY
SOLUTION PROPOSALS



Resource Hierarchy Solution

1. Index the resources (forks) from 1 to 5.
2. Each philosopher pick up the lower numbered fork first, and then the higher numbered fork.
3. After eating, put down forks in any order.

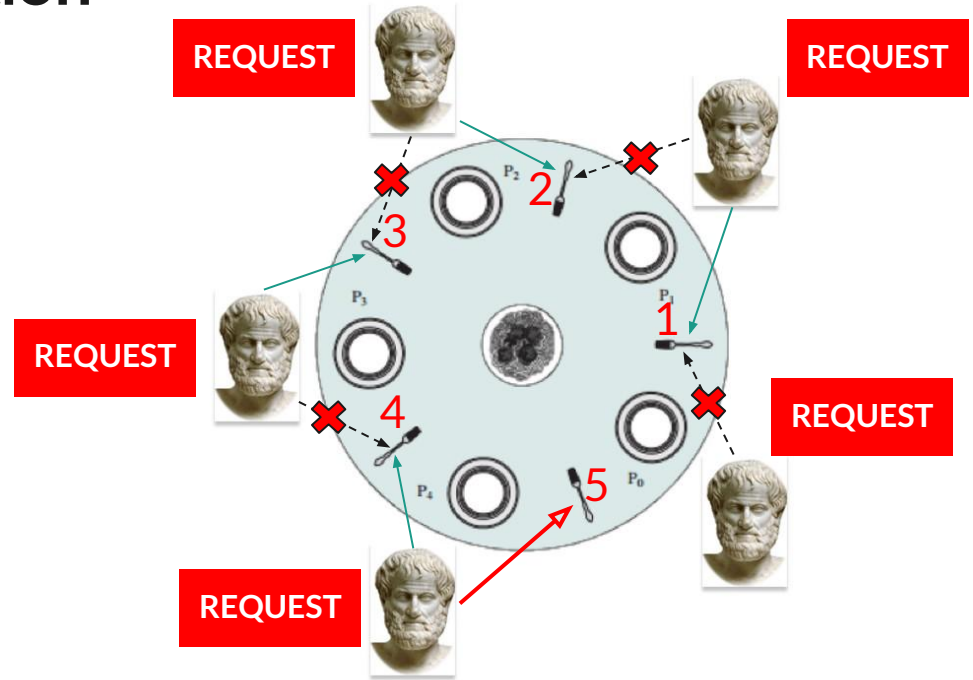


Resource Hierarchy Solution - Argument

01 philosopher request right fork first

NO "CIRCULAR WAIT"

DEADLOCK-FREE

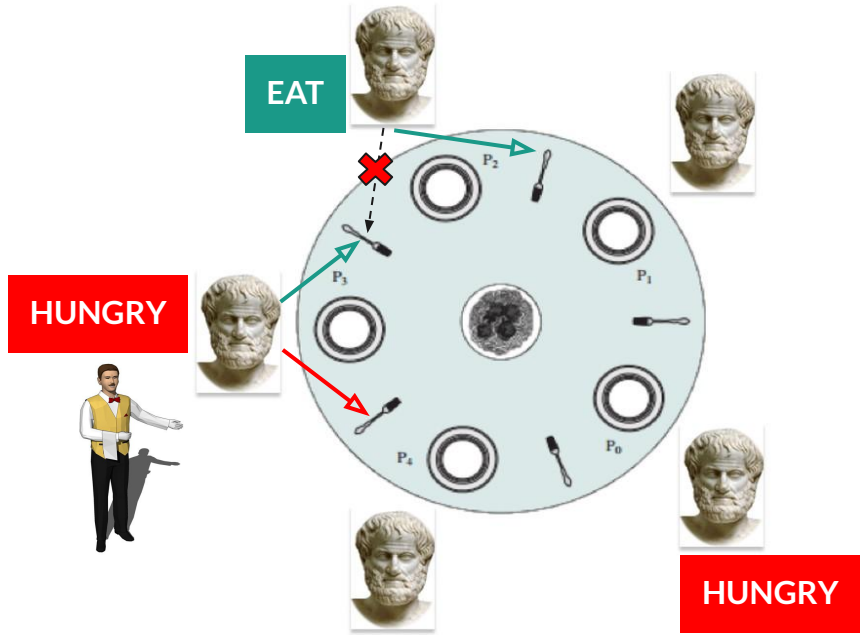


Resource Hierarchy Solution
- Drawback

01 philosopher requests forks at once…

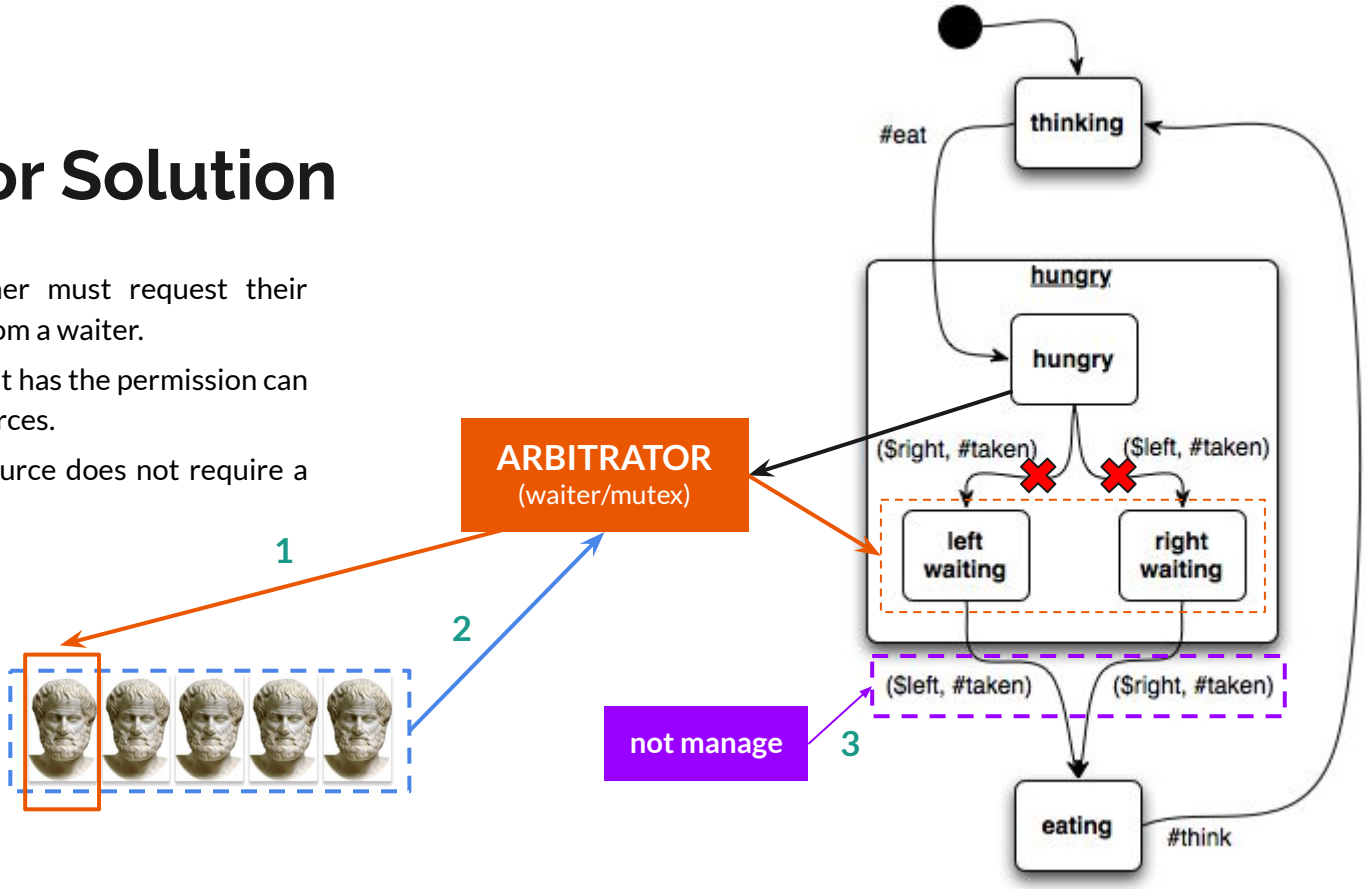
WHAT ABOUT THE OTHERS ?

STARVATION



Arbitrator Solution

1. Every philosopher must request their (shared) forks from a waiter.
2. Only the unit that has the permission can access the resources.
3. Releasing a resource does not require a permission.

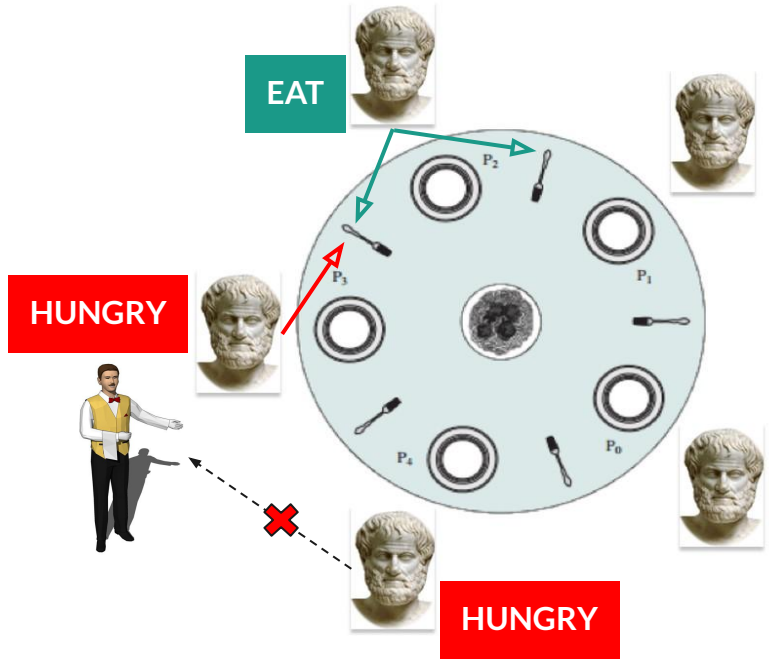


Arbitrator Solution - Argument

01 philosopher requests forks at once…

NO "CIRCULAR WAIT"

DEADLOCK-FREE

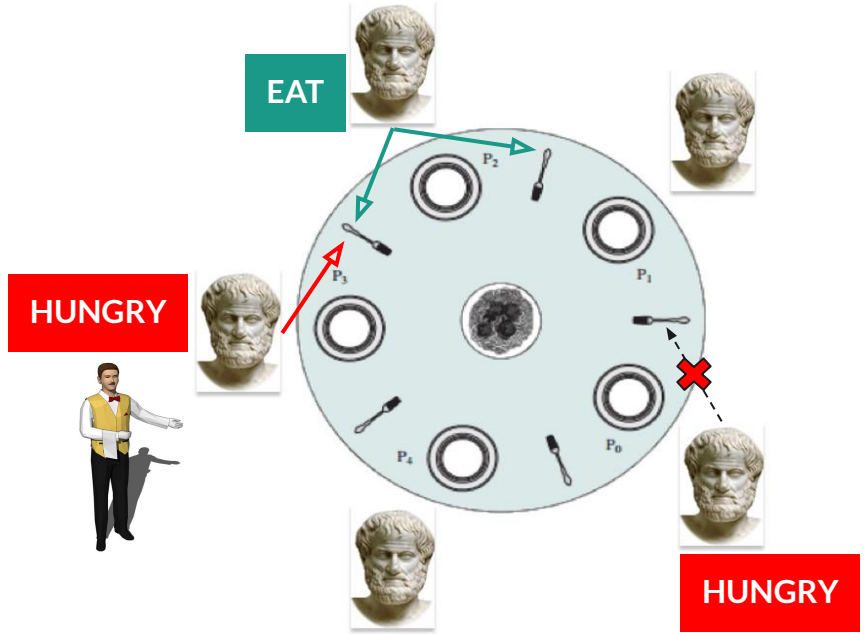


Arbitrator Solution - Drawback

01 philosopher requests forks at once…

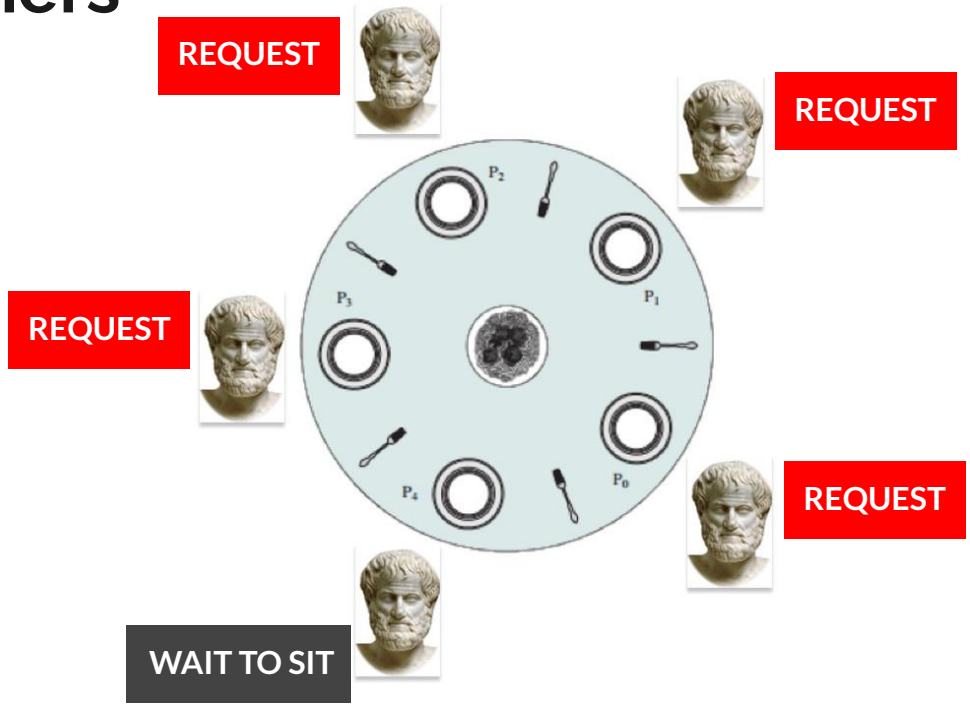
WHAT ABOUT THE OTHERS ?

PROGRESS



Limit the Number of Diners

1. Limiting the number of people requesting fork to 4
2. There exists one person who is not allowed to request fork
3. When at least one person finishes eating, the rest can request fork

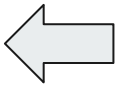


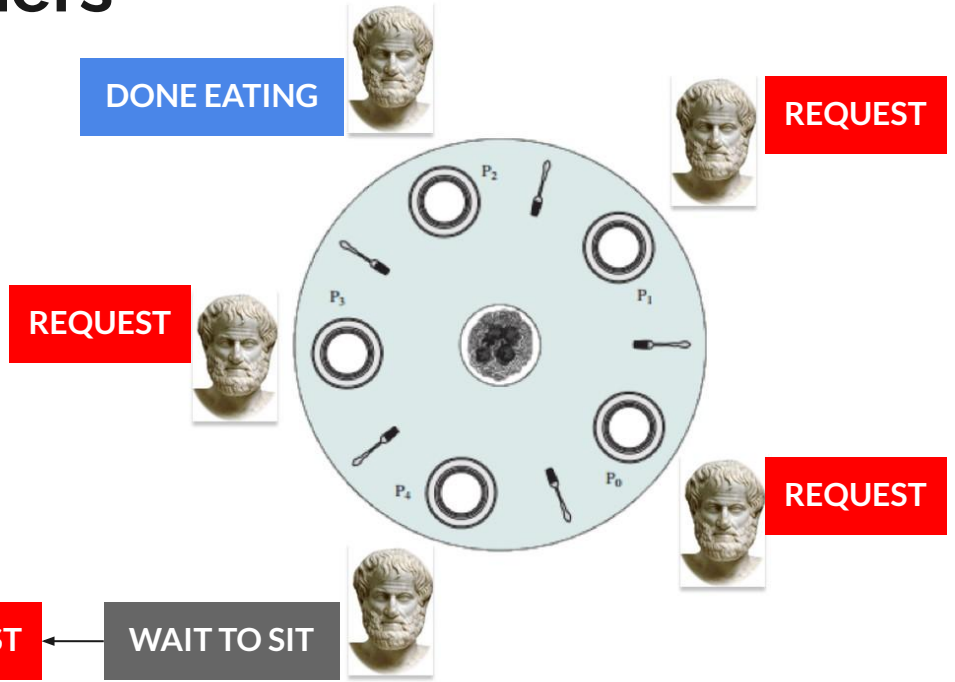
Limit the Number of Diners - Argument

04 philosopher requests forks at once…

NO “CIRCULAR WAIT”

DEADLOCK-FREE





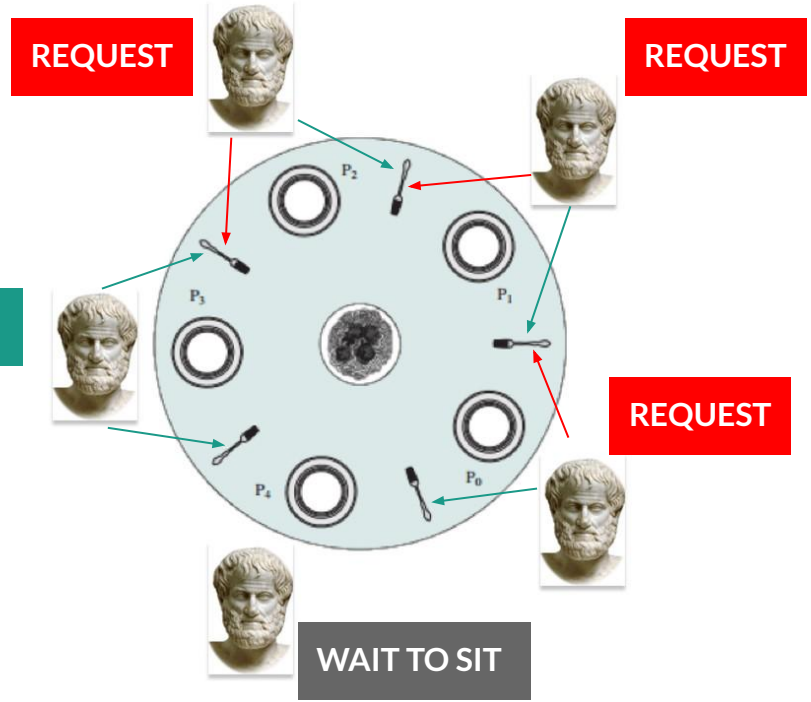
Limit the Number of Diners - Drawback

04 philosopher requests forks at once…

NO “CIRCULAR WAIT”

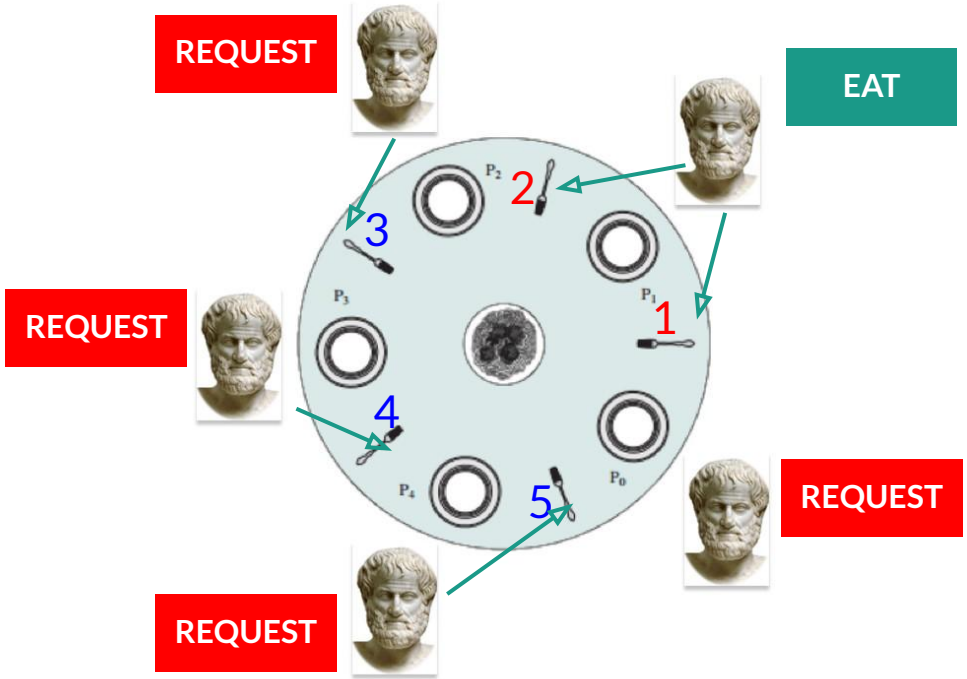
STARVATION

EAT



Chandy/Misra Solution

1. Each fork is either clean or dirty. Initially all forks are dirty.
2. When hungry, philosophers will request the left and right forks from neighbors.
3. When request received:
   - Eating: defer the request
   - The fork is dirty: clean the fork and pass it.
   - The fork is clean : defer the request

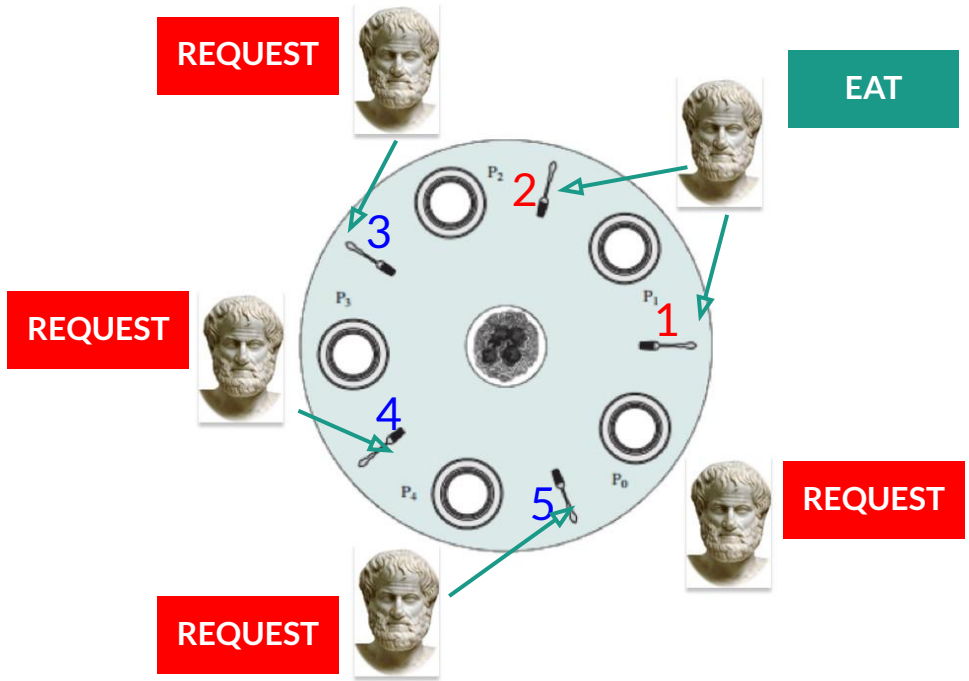


Chandy/Misra Solution - Argument

Each philosopher always get to eat...

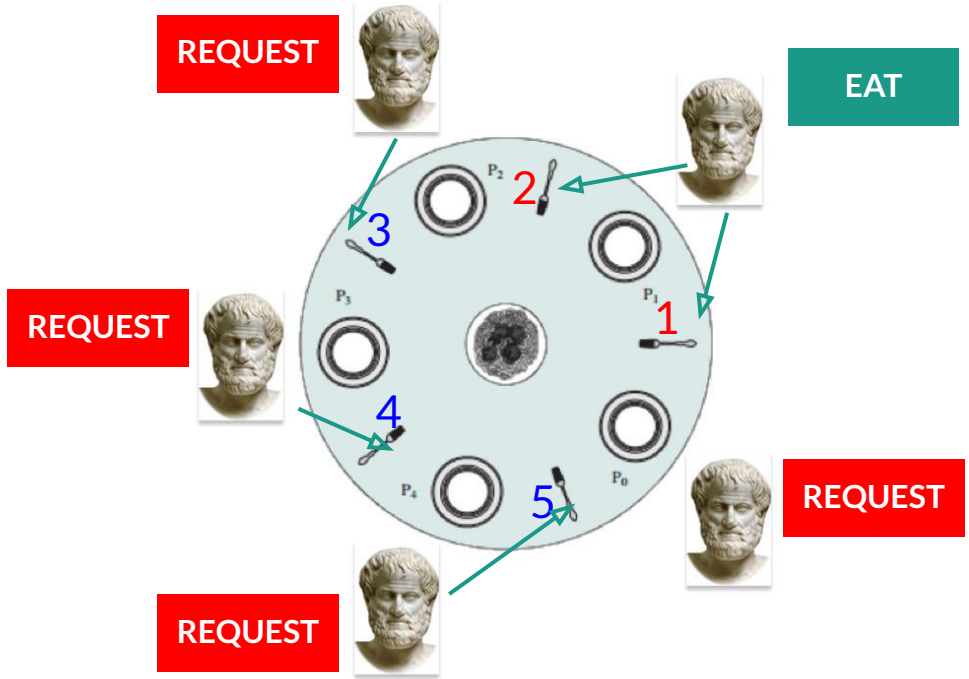
NO STARVATION

DEADLOCK-FREE

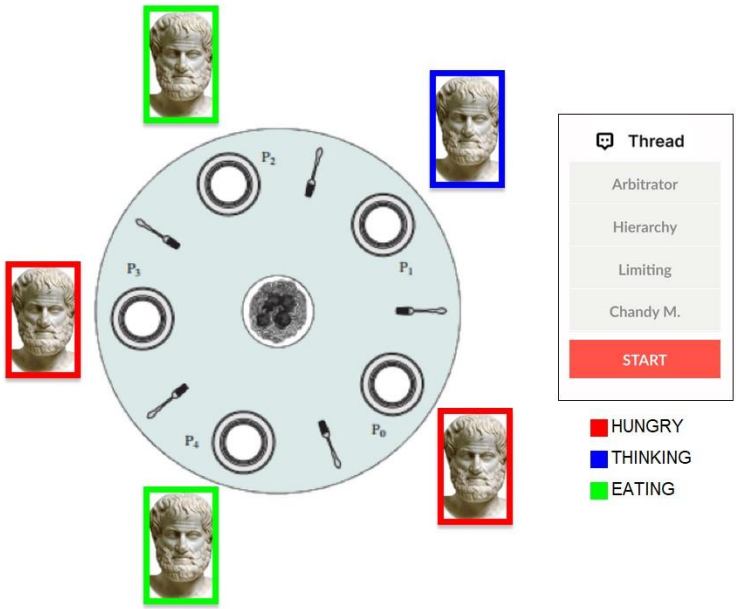


Chandy/Misra Solution - Drawback

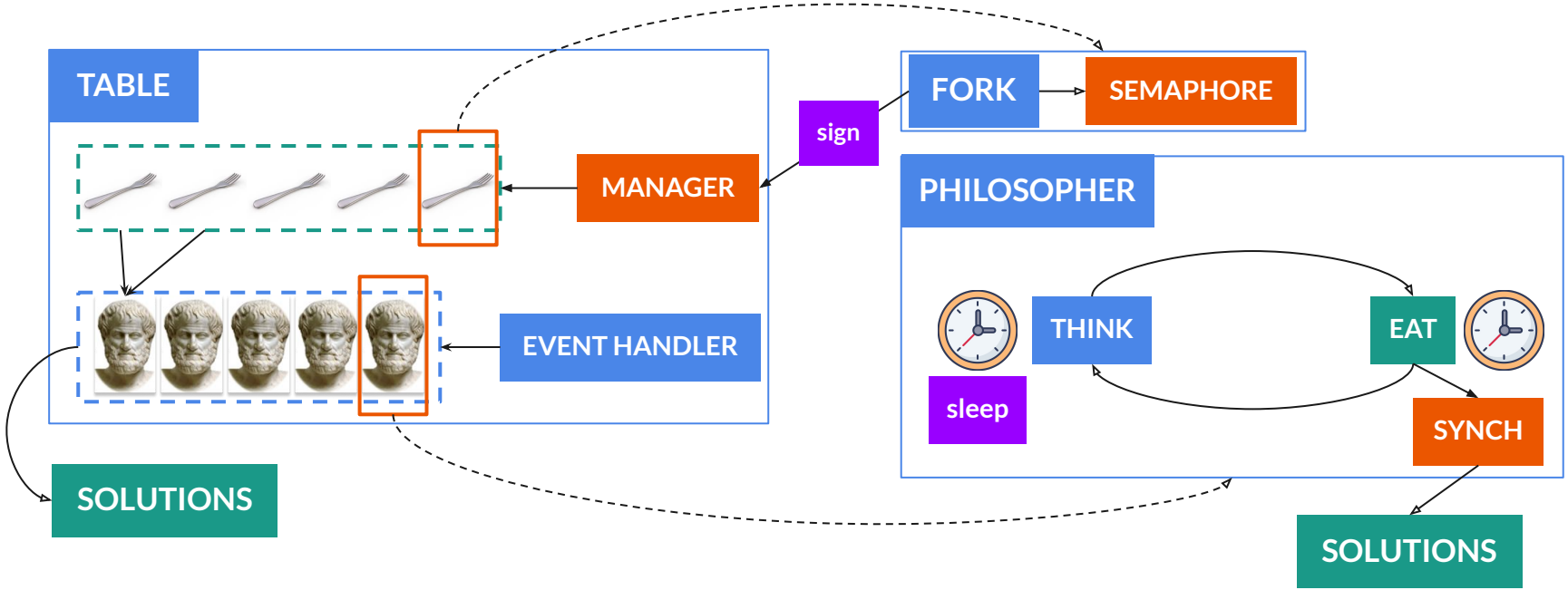
LONG WAIT CHAIN



PRACTICE
SOLUTION IMPLEMENTATION



Problem Modeling



Resource Hierarchy Solution

Algorithm 1 Resource hierachy [2]

```
semaphore array[0..4] fork ← [1, 1, 1, 1, 1]
  loop forever
    p1 : think
    p2 : wait(mutex)
    p3 : if philosopher.id = 5
    p4 :     wait(fork[(i + 1)%5])
    p5 :     wait(fork[i%5])
    p6 : else
    p7 :     wait(fork[i%5])
    p8 :     wait(fork[(i + 1)%5])
    p9 : end if
    p10: signal(mutex)
    p11: eat
    p12: signal(fork[i%5])
    p13: signal(fork[(i + 1)%5])
```

The fifth philosopher picks...

Right fork, then left fork

The others picks...

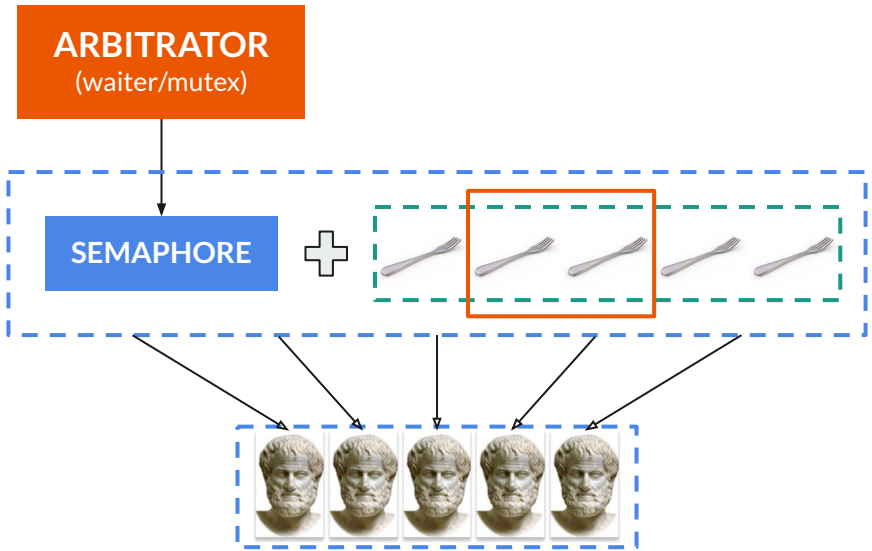
Left fork, then right fork

Arbitrator Solution

Algorithm 1 Arbitrator

semaphore $array[0..4] fork \leftarrow [1,1,1,1,1]$
  semaphore $mutex \leftarrow 1$
  loop forever
    p1 : think
    p2 : wait($mutex$)
    p3 : wait($fork[i]$)
    p4 : wait($fork[i+1]$)
    p5 : signal($mutex$)
    p6 : eat
    p7 : signal($fork[i]$)
    p8 : signal($fork[i+1]$)

from parent class



Limit the Number of Diners

Algorithm 2 Limit number of Diners

semaphore $array[0..4] fork \leftarrow [1, 1, 1, 1, 1]$
  semaphore $wait\_to\_sit \leftarrow 4$
  loop forever
    p1 : think
    p2 : wait($wait\_to\_sit$)
    p3 : wait($fork[i]$)
    p4 : wait($fork[i + 1]$)
    p5 : eat
    p6 : signal($fork[i]$)
    p7 : signal($fork[i + 1]$)
    p8 : signal($wait\_to\_sit$)

Maximum number of people request forks...

04 PHILOSOPHERS

A philosopher waits for at least...

01 PHILOSOPHER DONE EATING

Chandy/Misra Solution

Algorithm 4 Philosopher P_i

forks: $fork_l$, $fork_r$

loop forever

p1 : think

p2 : $fork_l.request$

p3 : $fork_r.request$

p5 : eat

p6 : $fork_l.finish$

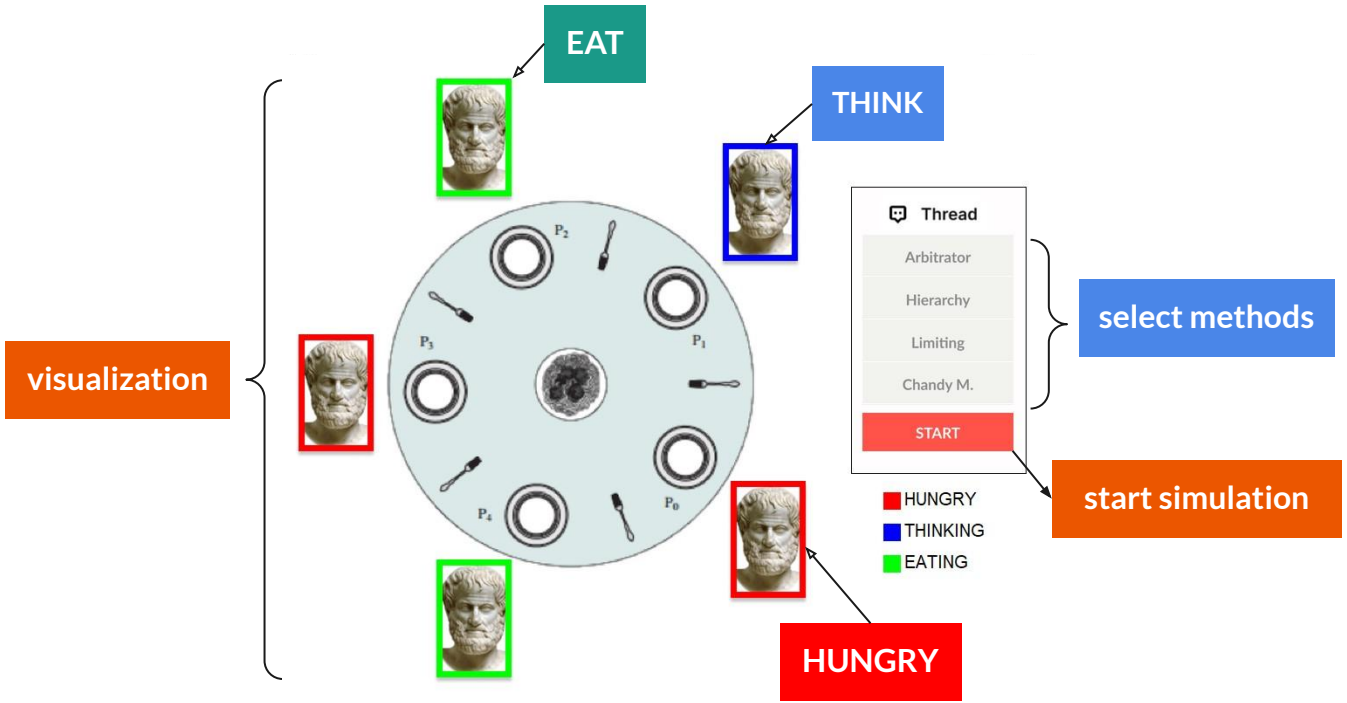
p7 : $fork_r.finish$

Know about...
BOTH FORKS

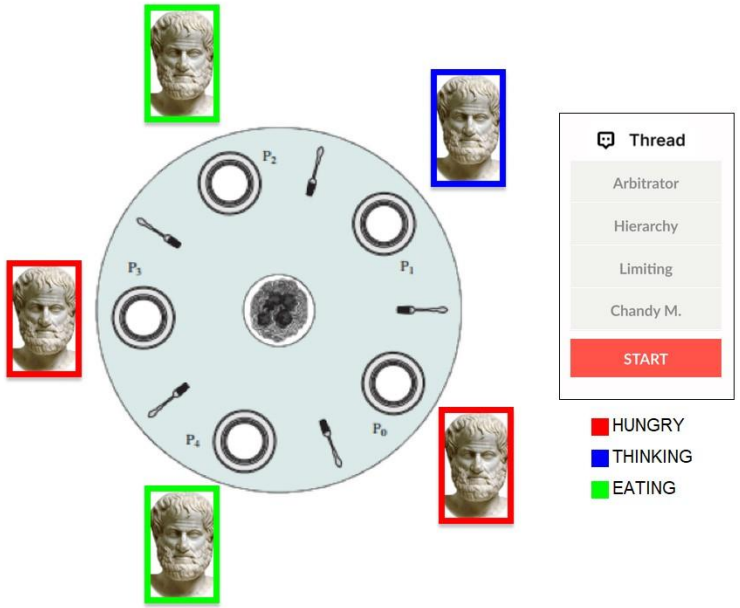
Philosopher is hungry...
CALL THE REQUEST ON THE FORKS

Done eating...
RELEASE THE FORKS

GUI


EAT
THINK
visualization
Thread
Arbitrator
Hierarchy
Limiting
Chandy M.
START
select methods
start simulation
HUNGRY
THINKING
EATING
HUNGRY

DEMO
THE PROGRAM



Operating Systems:
DINING PHILOSOPHERS

END

Nguyễn Tống Minh - 20204885
Nguyễn Công Đạt - 20200137
Ngô Thị Thu Huyền - 20200289
Nguyễn Trung Hiếu - 20204877

TEAM 16 present
SOICT (2022)

Header

A **paragraph** is defined as "*a group of sentences or a single sentence that forms a unit*" (Lunsford and Connors 116). Length and appearance do not determine whether a section in a paper is a paragraph. For instance, in some styles of writing, particularly journalistic styles, a paragraph can be just one sentence long.

Quote

Normal text...

"EMPHATIC TEXT"

AN OBJECT