CS & IT ENGINEERING

Operating System

Deadlock

Lecture -1

By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic Semaphore

Topic Questions on Semaphore

Topic Classical Problems on Synchronization

Topic ✓ Bounded Buffer Problem

Topics to be Covered

Topic Reader-Writer Problem

Topic Dining Philosopher Problem

Topic Deadlock

Reader-writer problem:-

A file is shared b/w multiple reader processes and multiple writer processes.

Topic : Reader-Writer Problem Solution

- If writer is accessing the file, then all other readers and writers will be blocked
- If any reader is reading, then other readers can read but writer will be blocked

	allowed	
currently	reader	writer
Reader	✓	×
writer	×	×

Topic : Reader-Writer Problem Solution

- Variables:
  - mutex: Binary Semaphore to provide Mutual Exclusion
  - wrt: Binary Semaphore to restrict readers and writers if writing is going on
  - Readcount: Integer variable, denotes number of active readers

- Initialization:
  - mutex: 1
  - w r t : 1
  - Read count: 0

Topic : Writer() Process

```
wait(wrt)
 // performs writing
signal(wrt)
```

Topic : Reader() Process

```
wait(mutex)
  readcount ++;
  if (readcount == 1)
    { wait(wrt);
    }
signal(mutex)
    // Performs Reading
wait(mutex)
  readcount --;
  if(readcount == 0)
  { signal(wrt); }
signal(mutex)
```

case 1 :-

A writer ① comes & writes

A reader ① comes ⇒ stuck at wait(wrt)

A reader ② comes ⇒ stuck at wait(mutex)

A writer ② comes

↳ stuck at wait(wrt)

wrt = ~~1~~ 0

mutex = ~~1~~ 0

readcount = ~~0~~ 1

Case 2:-

A reader① comes & reads

A writer comes ⇒ stuck at wait(wrt)

A reader② comes ⇒ Reads

A reader③ comes ⇒ Reads

Reader ② Exits ✓

Reader ① Exits ✓

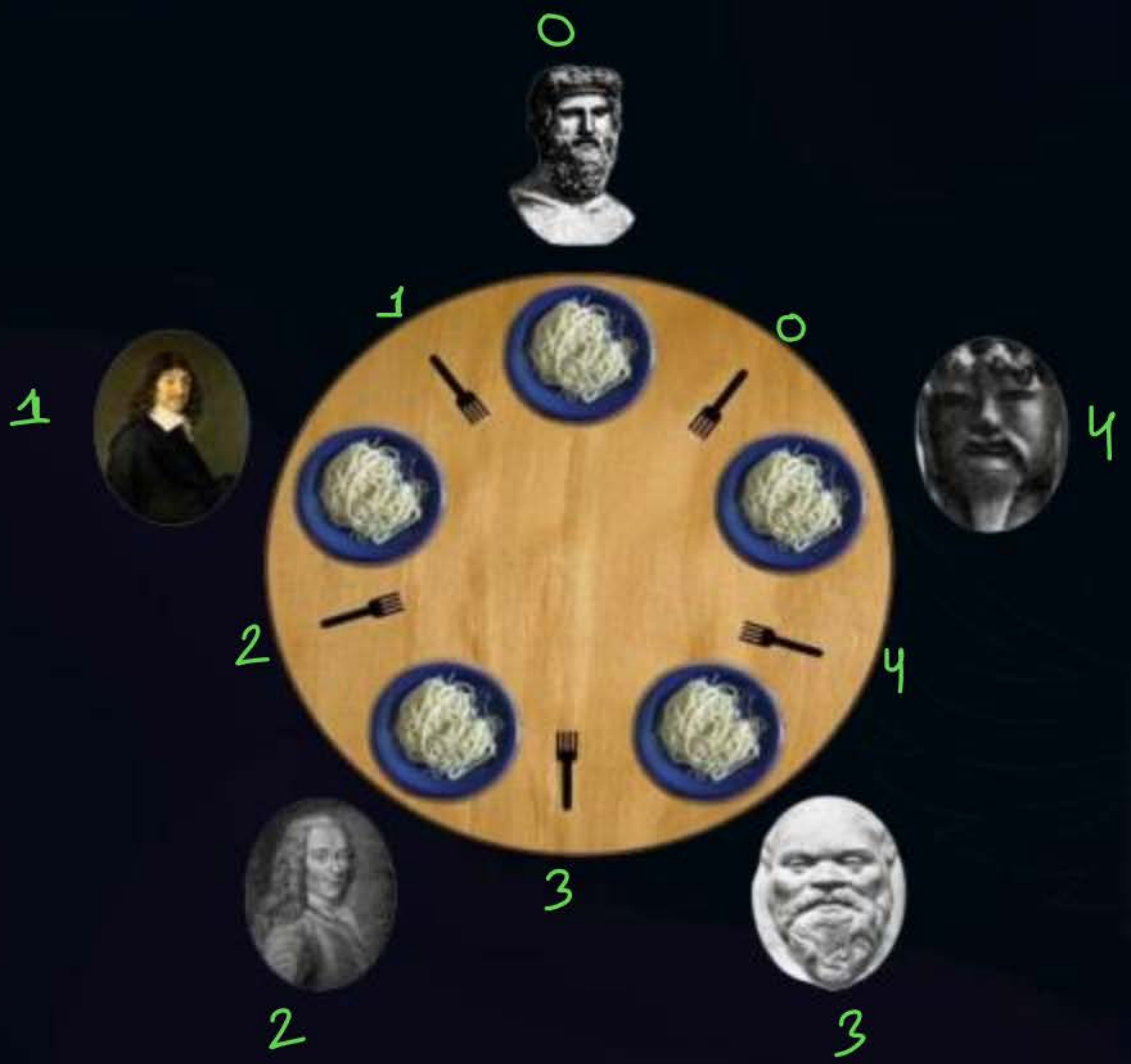
Reader ③ Exits ✓

wrt = ~~1~~ ~~0~~ 1

mutex = ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~0~~ 1 ~~1~~

readcount = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~2~~ ~~1~~ 0

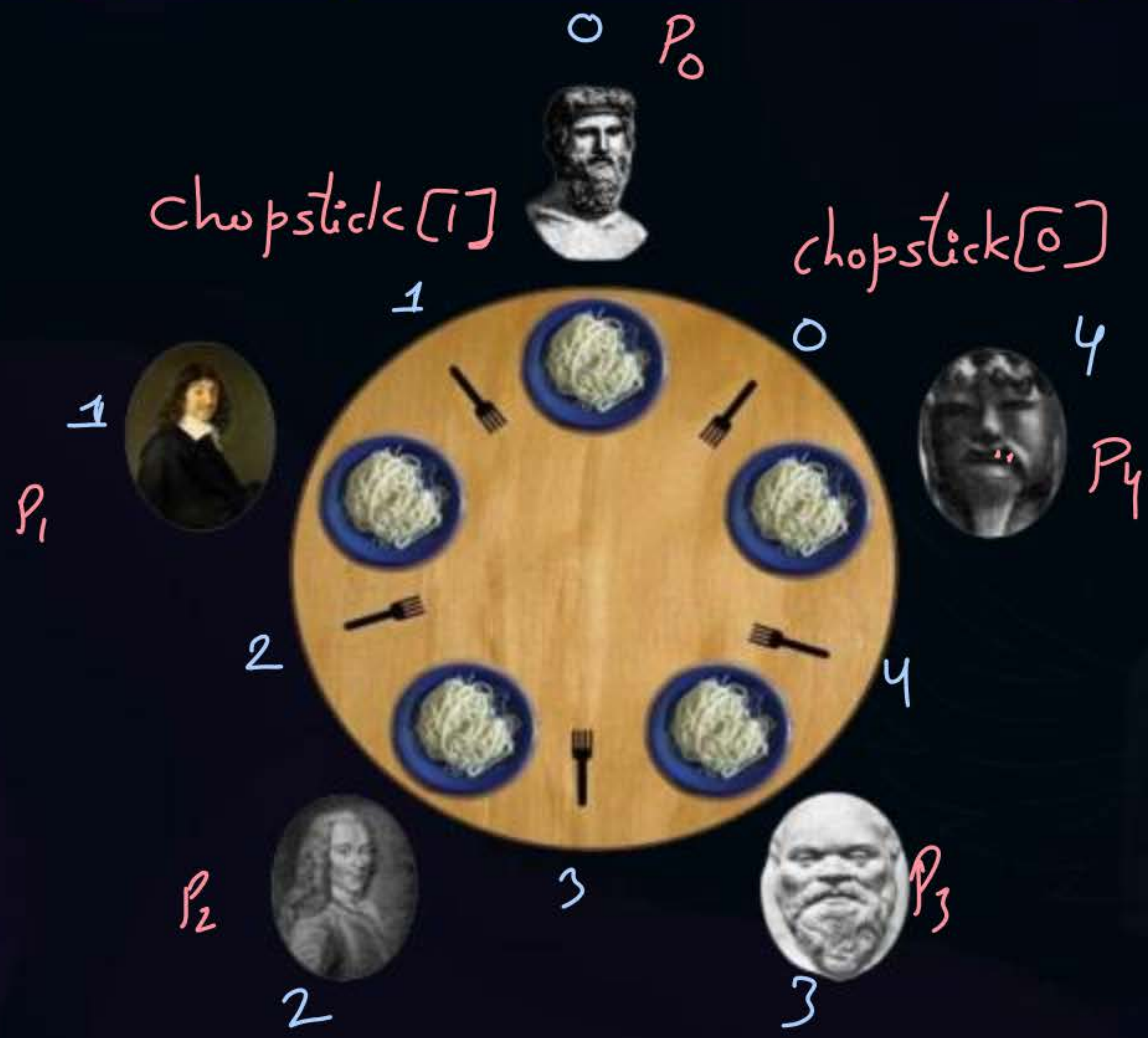
Topic : Dining Philosopher Problem



Topic : Dining Philosopher Problem

- K philosophers seated around a circular table
- There is one chopstick between each philosopher
- A philosopher may eat if he can pick up the two chopsticks adjacent to him
- One chopstick may be picked up by any one of its adjacent followers but not both

Topic : Dining Philosopher Problem Solution



Topic : Dining Philosopher Problem Solution

for k number of chopsticks, consider an array of binary semaphores of size k.

chopstick[k] = {1, 1, 1, ..., 1};

each philosopher runs a process:-

$P_0, P_1, \ldots, P_{k-1}$

Topic : Dining Philosopher Problem Solution

```
              Pi
---------------------------------
wait(chopstick[i])
wait(chopstick[(i+1) % k])

   //eat
signal(chopstick[i])
signal(chopstick[(i+1) % k])
```

It can suffer from deadlock.

$\Downarrow$

if one-by-one all philosophers pick one chopstick.

Topic : Dining Philosopher Problem Solution

Some of the ways to avoid deadlock are as follows –

1. There should be at most (k–1) philosophers on the table

Topic : Dining Philosopher Problem Solution

Some of the ways to avoid deadlock are as follows –

1. There should be at most (k–1) philosophers on the table
2. A philosopher should only be allowed to pick their chopstick if both are available at the same time

Topic : Dining Philosopher Problem Solution

Some of the ways to avoid deadlock are as follows –

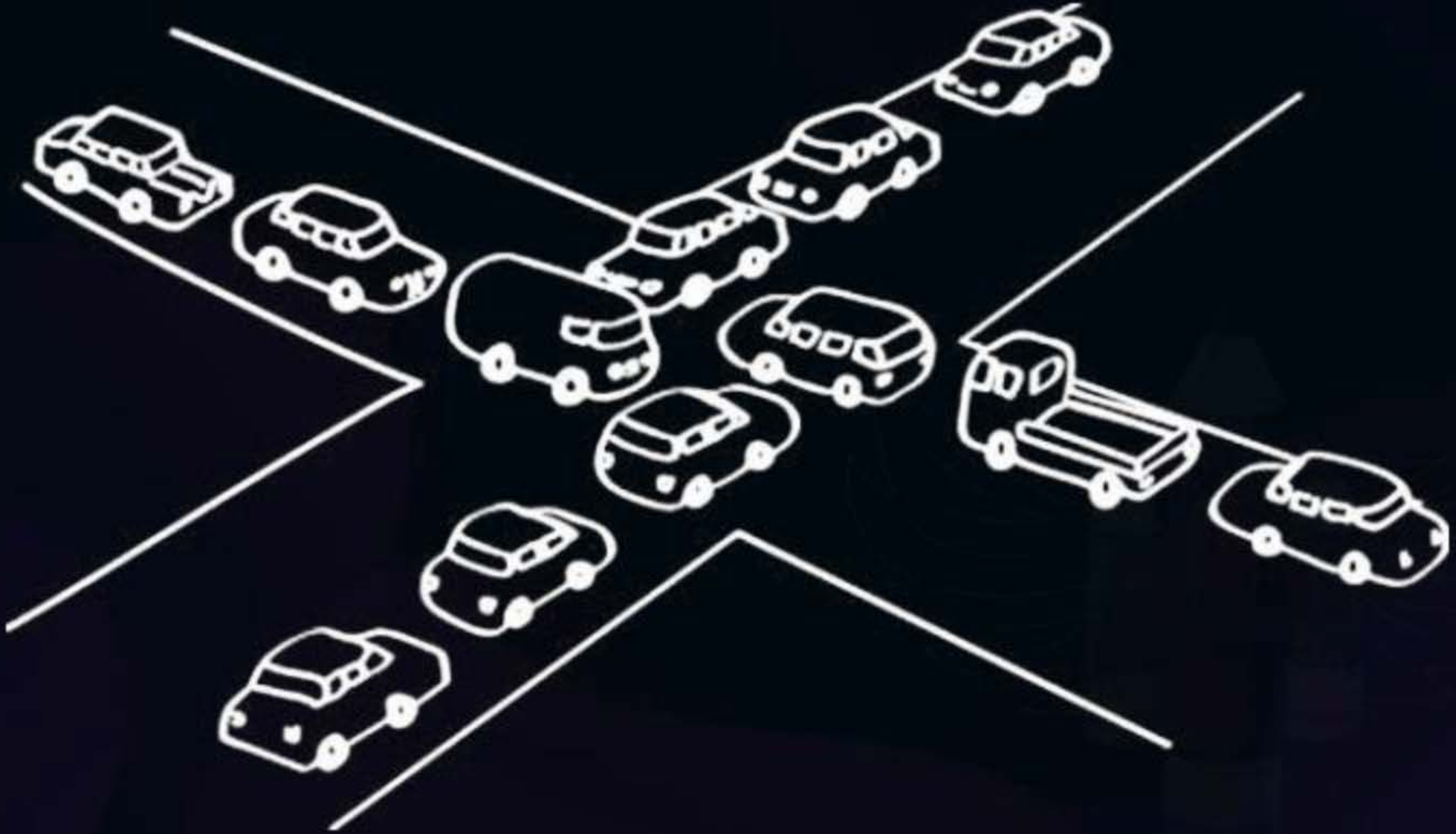
1. There should be at most (k–1) philosophers on the table
2. A philosopher should only be allowed to pick their chopstick if both are available at the same time
3. One philosopher should pick the left chopstick first and then right chopstick next; while all others will pick the right one first then left one

$\hookrightarrow$ asymmetry in picking up chopstick

Topic : Deadlock

If two or more processes are waiting for such an event which is never going to occur

Topic : Deadlock



Topic : Operations on Resources

3 Operations on resources:

1. Request :- A process can request to OS to allocate a resource to it.
2. Use
3. Release

4. Wait :- If resource is not available then process will wait.

2 mins Summary

Topic Reader-Writer Problem

Topic Dining Philosopher Problem

Topic Deadlock

Happy Learning

THANK - YOU