Parallel Programming

FALL 2023

Time-Out Strategy

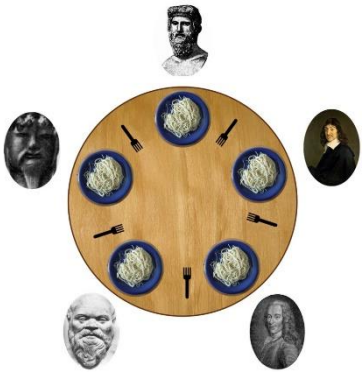
Group Turan's

Murat Turan - 200316022

Cevdet Ahmet Turan - 200316013

25 December 2023

Assoc. Prof. Dr. Bora CANBULA



INTRODUCTION

The Dining Philosophers problem is a classic synchronization and concurrency problem that illustrates issues with resource sharing and deadlock avoidance in concurrent systems. It involves a scenario where a group of philosophers sits around a dining table, each with a plate of spaghetti and a fork between every pair of plates. However, to eat, a philosopher needs two forks, one for each hand, and may only pick up the forks adjacent to their plate.

Timeout Strategy:

In addressing the Dining Philosophers problem, one effective strategy is the timeout approach. The timeout strategy introduces a time-bound mechanism for philosophers attempting to acquire the necessary resources (forks) for eating.

Implementation:

Philosopher's Behavior:

- Philosophers attempt to acquire both their left and right forks simultaneously.
- A designated time limit (timeout) is set for this acquisition process.

Acquisition Outcome:

- If a philosopher successfully acquires both forks within the specified time, they proceed to eat.
- Eating involves a decrease in the spaghetti count and a simulated eating duration.

Timeout Handling:

- If a philosopher cannot obtain both forks within the defined time limit, they release any acquired forks without eating.
- This mechanism prevents philosophers from waiting indefinitely and ensures fairness in resource access among the philosophers.

Purpose of Timeout Strategy:

The timeout strategy aims to mitigate potential deadlocks by introducing a time-bound criterion for resource acquisition. It prevents a philosopher from indefinitely waiting for a fork, thereby averting a scenario where all philosophers hold one fork and wait indefinitely for the second one, causing a deadlock.

Significance:

The timeout strategy serves as an essential method to prevent resource contention and deadlock situations in systems with shared resources. By imposing time constraints on resource acquisition attempts, this strategy promotes fairness and ensures progress, crucial for avoiding deadlock scenarios like the Dining Philosophers problem.

Code Explanation:

Class Definitions: Fork and Philosopher

- **Fork Class:** This class represents the forks used by the philosophers during their dining process. It includes methods to manage the state of the forks and track their ownership by the philosophers.
- **Philosopher Class:** Represents individual philosophers as threads. Each philosopher engages in thinking and eating activities. This class manages the philosopher's behavior, such as thinking time, eating time, and interactions with the forks.

Timeout Strategy Implementation in eat_timeout Method

- The Philosopher class implements the eat_timeout() method, crucial for tackling the Dining Philosophers problem using a timeout strategy.
- Inside the eat_timeout() method, a philosopher attempts to acquire both the left and right forks simultaneously within a defined time limit (timeout).
- If successful in acquiring both forks within the specified time, the philosopher proceeds to eat by decreasing the spaghetti count and simulating the eating duration.
- However, if the philosopher cannot acquire both forks within the timeout duration, they release any acquired forks without eating, enabling other philosophers to attempt acquiring the forks.

Summary: The provided code offers a solution to the Dining Philosophers problem utilizing a timeout strategy. It demonstrates how philosophers attempt to acquire forks within a specified time limit, ensuring they do not get trapped in a deadlock scenario. This code effectively illustrates the interaction between philosophers, forks, and the prevention of potential deadlocks by employing timeouts.