Vishwakarma Institute of Technology, Pune
Department Of Electronics & Telecommunication Engineering

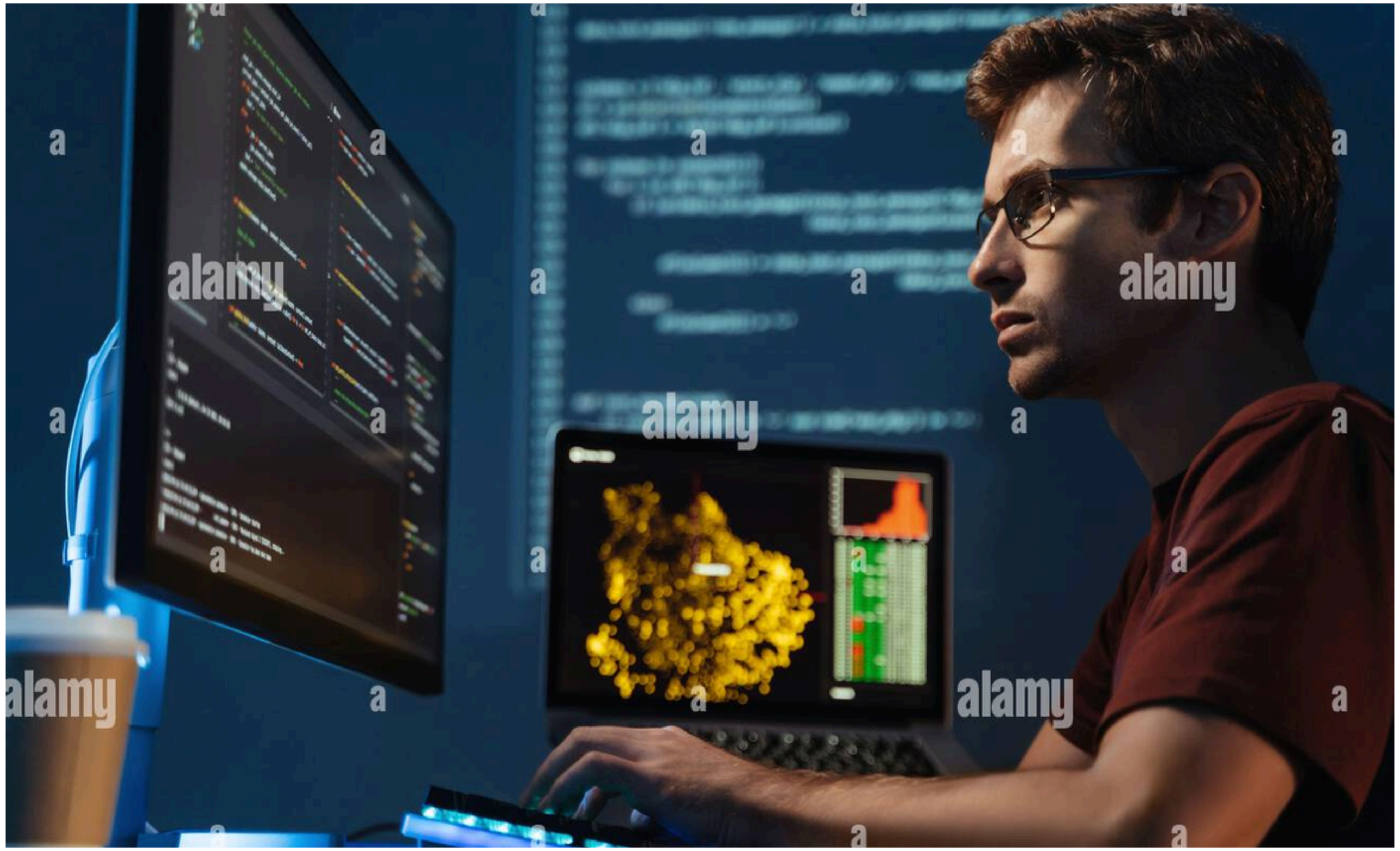
Comparative Analysis of Deadlock Detection Algorithms

Guided By : Prof. Ashwini Pulujkar

9	Sakshi Rathod	12210324	Division:C
10	Samruddhi Raut	12210307	Batch:1
21	Ruta Sapate	12210930	Group:7

Introduction to Deadlock Detection

Deadlock detection is a technique used in operating systems to identify and resolve situations where multiple processes are stuck in a waiting state, unable to proceed. It is a crucial aspect of resource management and ensures that the system remains operational and efficient.



Necessary Conditions for Deadlock

Deadlock can arise if the following four conditions hold simultaneously (Necessary Conditions)

- Mutual Exclusion: Two or more resources are non-shareable (Only one process can use at a time).
- Hold and Wait: A process is holding at least one resource and waiting for resources.
- No Preemption: A resource cannot be taken from a process unless the process releases the resource.
- Circular Wait: A set of processes waiting for each other in circular form.

Deadlock Detection Algorithms

Deadlock detection algorithms systematically analyze the resource allocation state to identify deadlocks. Algorithms can either use a resource allocation graph or a wait-for graph, which depicts processes waiting for each other.

01

Resource Allocation Graph

This method involves searching for cycles in the resource allocation graph to detect deadlocks.

02

Wait-For Graph

This approach uses a graph to represent the waiting relationships between processes, where cycles indicate a deadlock.

03

Matrix Representation

This approach uses matrices to track resource allocation and process requests, and algorithms can analyze the matrix to detect deadlocks.

Banker's Algorithm

The Banker's Algorithm is a deadlock avoidance technique that ensures the system remains in a safe state, where all processes can be completed without encountering a deadlock. It requires detailed information about the system's available resources and the maximum resource needs of each running process.

Processes	Allocation	Max	Available
	A B C	A B C	A B C
P0	2 1 0	8 6 3	4 3 2
P1	1 2 2	9 4 3	
P2	0 2 0	5 3 3	
P3	3 0 1	4 2 3	

01
Resource Allocation
The algorithm keeps track of the current allocation of resources to each process, as well as the maximum resource needs of each process.

02
Safety Check
Before granting a resource request, the algorithm checks if granting the request would put the system in an unsafe state, where a deadlock could occur

03
Deadlock Avoidance
If granting the request would make the system unsafe, the request is denied to avoid the potential deadlock.

Overview of Deadlock Detection Algorithms in Blockchain Systems

1. Push-Relabel Algorithm

- Finds routes for payments in a network, managing how funds flow between nodes
- Routes payments through multiple paths, allowing concurrent transactions without constraint violations.

2. Fulgor Algorithm

- Resolves deadlocks and ensures privacy in payment networks.
- Manages transaction order and uses off-chain channels to avoid immediate blockchain commitment.

.

3. Scalable Algorithm

- Detects deadlocks in distributed systems with many threads.
- Monitors out-of-order operations with a runtime complexity of O(1), handling large-scale thread interactions.

Overview of Deadlock Detection Algorithms in Blockchain Systems

4. Knapp’s Algorithm

- Detects deadlocks using various techniques like diffusion computation and global state detection.
- Utilizes a wait-for graph and checks system states to identify deadlock conditions.

5. Consensus Algorithm

- Ensures agreement among nodes on the state of the blockchain.
- Validators and miners collaborate to add new blocks, using protocols like Proof of Work or Proof of Stake (e.g., Tendermint).

ALGORITHM	PROS	CONS
Fulgor Algorithm	– Identifies pre-lock edges efficiently – Maintains privacy and progress.	– Complex to implement. – Limited by network capacity.
Push-Relabel Algorithm	– Optimizes route selection in payment networks. – Handles concurrent executions well.	– Needs global knowledge for routing. – Complexity grows with network size.
Consensus-Based Algorithm	– Ensures security and trust via decentralized consensus. – No single point of failure.	– High communication and computation overhead. – Scalability issues in large networks.
Scalable Deadlock Detection Algorithm	– Efficient in large-scale systems. – O(1) runtime for collective operations.	. – Complex to implement and tune. – Requires advanced communication systems.
Knapp's Algorithm	– Versatile with multiple detection techniques. – Uses global state for thorough analysis.	– Complex due to multiple methods. – Overhead in state management.

Why Fulgor's Algorithm is efficient

- Privacy and Progress: Effectively manages privacy and ensures transaction progress.
- Versatility: Performs well in varied environments, both controlled and uncontrolled.
- Efficiency: Handles complex networks and payment routes efficiently.

Reasons for Preference:

- Privacy and Scalability: Excels in maintaining privacy and handling scalability in blockchain networks.
- Comprehensive Functionality:Balances efficiency and privacy, crucial for modern systems.
- Adaptability: Suitable for both small and large-scale deployments.

Comparison of Traditional vs. Modern Deadlock Detection Algorithms

ASPECT	TRADITIONAL ALGORITHMS	MODERN ALGORITHMS
Examples	Banker's Algorithm, Resource Allocation Graph (RAG), Wait-For Graph (WFG)	Fulgor Algorithm, Push-Relabel Algorithm, Consensus Algorithms, Scalable Deadlock Detection, Knapp's Algorithm
Scalability	- Often struggle with large systems. - Performance issues in high-load scenarios.	- Designed to handle large-scale and high-throughput environments effectively. - Better scalability for distributed systems.
Dynamic Systems	- Less effective in dynamic environments where processes and resources frequently change.	- More adaptable to dynamic and changing environments. - Suitable for distributed and decentralized systems.
Complexity	- Generally simpler and easier to understand and implement.	- More complex and can require advanced setup and understanding.

ASPECT	TRADITIONAL ALGORITHMS	MODERN ALGORITHMS
Resource Requirements	– Can be resource-intensive in large systems.	– Often optimized for performance and may use advanced techniques to reduce overhead.
Decentralization	– Not well-suited for decentralized systems.	– Designed for decentralized and distributed environments.
Deadlock Prevention vs Detection	– Mainly focused on prevention centralized detection	– Focuses on detection and handling of deadlocks in distributed systems
Security & Privacy	-Limited focus on security & privacy	-Often includes consideration for security & privacy (e.g. Fulgar's focus on privacy)

Thank You