Byzantine Failures in Distributed Systems

Trust and Resilience in Distributed Systems

Burak Metehan Tunçel
tuncel.burak@metu.edu.tr

Wireless Systems, Networks and Cybersecurity Laboratory
Department of Computer Engineering
Middle East Technical University
Ankara Turkey

March 28, 2024

Outline of the Presentation

Agenda

Clock Synchronization

Clock Synchronization in Distributed Systems

Ensuring accurate and consistent timekeeping across geographically distributed systems is fundamental for coordinated operation.
Traditional clock synchronization algorithms assume processes behave correctly and exchange reliable information; however, real-world systems are not immune to failures. In some scenarios, processes may exhibit malicious or arbitrary behavior.
Therefore, developing robust clock synchronization algorithms that can tolerate these failures is crucial for ensuring the reliability and integrity of distributed systems.

Agenda

What is the solution/contribution

- Robust clock synchronization algorithms tolerating Byzantine failures, which exhibit malicious or arbitrary behavior, is crucial.
- **The Mahaney-Schneider synchronizer** stands out as a practical and efficient solution for achieving this goal.
- Leverageing message exchange and statistical analysis to filter out misleading information.
- Converging towards a reliable estimate of the system time, even in the presence of Byzantine processes.

Agenda

Motivation/Importance

- **Trust in Decentralized Systems:** BFT is essential for establishing trust in decentralized systems like blockchain, ensuring operations continue even when some participants are unreliable.
- **Real-World Applications:** It's crucial for safety in critical applications such as aviation and autonomous vehicles, where failure can have dire consequences.
- **Philosophical Reflection:** BFT prompts reflection on trust and consensus in systems where misinformation or malice is possible, highlighting its broader implications.
- **Evolving Distributed Systems:** As these systems grow in scale and importance, the role of BFT in ensuring resilience against problematic conditions becomes increasingly vital.

Agenda

Definitions

- Byzantine failures are the scenarios where processes exhibit arbitrary behavior including sending incorrect information, withholding data, or even actively trying to disrupt the synchronization process.
- Byzantine Fault Tolerance (BFT) is a property of distributed systems that allows them to continue operating correctly even when some of the nodes fail in arbitrary or malicious ways.

Background

- Clock synchronization in distributed systems ensures consistent timekeeping across geographically.
- Traditional algorithms assume well-behaved processes such as Lamport's logical clocks and Cristian's algorithm.
- Traditional algorithms fail with Byzantine failures.

Background

- BFT protocols can tolerate Byzantine failures but are computationally complex and expensive.
- The Mahaney-Schneider synchronizer is a BFT clock synchronization algorithm that is efficient and robust.
- Work even in the presence of Byzantine failures.

Agenda

Main Point 1

TODO: Complete after code implementation and experiments.

Agenda

Main Result 1

TODO: Complete after code implementation and experiments.

Agenda

Conclusions

TODO: Complete after code implementation and experiments.

References

Questions

THANK YOU

Byzantine Failures in Distributed Systems
Trust and Resilience in Distributed Systems

presented by Burak Metehan Tunçel
tuncel.burak@metu.edu.tr

March 28, 2024

Mutual Exclusion in Distributed Systems

Balancing Efficiency and Safety in Distributed Systems

Burak Metehan Tunçel
tuncel.burak@metu.edu.tr

Wireless Systems, Networks and Cybersecurity Laboratory
Department of Computer Engineering
Middle East Technical University
Ankara Turkey

March 28, 2024

Outline of the Presentation

Agenda

The problem

Tell a STORY from the background to the conclusion

The Problem Name

- Ensuring data consistency and preventing race conditions is crucial. When multiple processes attempt to access and modify shared resources simultaneously, some unpredictable outcomes and data corruption can be experienced.
- Mutual Exclusion (ME) algorithms dictate the order in which processes interact with critical sections.
- By ensuring only one process executes within a critical section at a time, ME algorithms uphold data integrity.
- There is need for a ME algorithm that is deadlock-free, message-efficient.

Agenda

What is the solution/contribution

- The realm of Mutual Exclusion (ME) algorithms in distributed systems boasts a rich history.
- Each iteration building upon the strengths and addressing the limitations of its predecessor.
- The Agrawal-El Abbadi algorithm builds upon these existing solutions, aiming to achieve a balance between message complexity and deadlock freedom.
- Introduces the concept of quorums, subsets of processes that must grant permission for a process to enter the critical section.
- Reduces message overhead compared to algorithms requiring communication with all processes.

Agenda

Importance of the Problem

- **Data Integrity:** Ensures only one process can modify a shared resource at a time, maintaining data accuracy.
- **Resource Allocation:** Manages limited resources efficiently among processes, avoiding starvation.
- **Avoiding Deadlocks:** Prevents or resolves deadlock situations, where processes wait indefinitely for resources.
- **System Reliability:** Contributes to stable system operations by managing access to shared resources.

Motivation Behind the Agrawal-El Abbadi Algorithm

- **Efficiency:** Aims to reduce communication overhead, making the system more efficient.
- **Scalability:** Designed to handle increasing numbers of processes and resources effectively.
- **Deadlock Prevention:** Focuses on eliminating or easily resolving deadlocks, improving system robustness.
- **Practicality:** Provides a straightforward and effective approach for real-world application

Agenda

Some Terminology

- **Mutual Exclusion (ME):** Prevents simultaneous access to a shared resource by multiple processes to avoid data inconsistency.
- **Critical Section:** A part of the system where access must be exclusive to prevent race conditions.
- **Message Complexity:** The number of messages needed to perform a task in a distributed system, with a goal to minimize for efficiency.
- **Deadlocks:** A standstill where processes wait indefinitely for each other to release resources.
- **Timestamps:** Used to order events or messages in a distributed system, ensuring a consistent sequence.
- **Quorums:** A selected group of processes whose approval is required for a process to access the critical section, reducing the need for full consensus.
- **Broadcasts:** Sending a message to all processes in the system, used for consistency but can increase message overhead.

Background

The journey towards efficient mutual exclusion (ME) in distributed systems has seen significant milestones, each addressing the challenges of maintaining data consistency, reducing message complexity, and preventing deadlocks.

- Lamport's Bakery Algorithm (1978):
  - **Overview:** Introduced a ticket-based system where processes obtain numbers to ensure orderly access to the critical section.
  - **Pros:** Simple and easy to implement.
  - **Cons:** High message overhead due to system-wide broadcasts.
- Ricart-Agrawala Algorithm (1981):
  - **Overview:** Utilizes timestamps in request messages, granting access based on the precedence of timestamps.
  - **Pros:** Reduces message complexity compared to Lamport's algorithm.
  - **Cons:** Risk of deadlocks due to potential circular waiting.

Background

- Maekawa's Voting Algorithm (1985):
  - **Overview:** Implements a voting mechanism among processes, allowing entry to the critical section through a virtual election.
  - **Pros:** Eliminates deadlocks.
  - **Cons:** Can result in high message overhead in larger systems.
- Towards the Agrawal-El Abbadi Algorithm:
  - Building on these foundational works, the Agrawal-El Abbadi algorithm introduces quorums -specific subsets of processes- to grant access to the critical section, aiming for an optimal balance between message efficiency and deadlock prevention.

Agenda

Main Point 1: A Figure

Abstract the Major Results

TODO: Complete after code implementation and experiments.

Agenda

Main Result 1

TODO: Complete after code implementation and experiments.

Agenda

Conclusions

TODO: Complete after code implementation and experiments.

References

Questions

THANK YOU

Mutual Exclusion in Distributed Systems

Balancing Efficiency and Safety in Distributed Systems

presented by Burak Metehan Tunçel
tuncel.burak@metu.edu.tr

March 28, 2024