# Towards Distributed Software Transactional Memories

**Paolo Romano**
INESC – ID, Lisbon, Portugal

*joint work with:* Nuno Carvalho, Maria Couceiro, Luís Rodrigues

*FCT / Universidade Nova de Lisboa*
*1 April, 2009*

# Roadmap

- (Software) Transactional Memories
  - What, how and why?
- FénixEDU & the PASTRAMY Project
- Overview of Database Replication Schemes
- STM Replication:
  - Critical issues
  - Some of our current research lines
- Conclusions

# (Software) Transactional Memory: What, how, why ?

## What is it?

(Software) tool aimed at simplifying development of concurrent programs by leveraging on the abstraction of atomic, isolated transactions.

## How is it achieved?

Transparently detecting conflicting memory accesses and aborting non-serializable transactions.

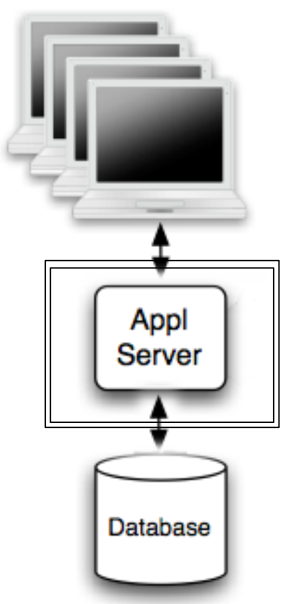
## Why all this ado of late?

Multi-core CPUs are nowadays mainstream, amplified interest in easing parallel programming

# The FénixEDU System

- Open source project supporting a wide range of activities of an university e-campus:
  - already used in ~10 universities
  - three-tier J2EE Web application
  - <u>**first production system to rely on STMs**</u>
- Real-life system raising challenging research issues!

# High level
# FénixEDU Architecture




**Application Server**

- in-memory object oriented domain model
- synchronization of concurrent transactions via **JVSTM**:
  - a multi-versioned, lock-free STM

**Current System Bottleneck**

**Ba base**

- ensures data durability
- overcomes appl. servers' memory capacity constraints

# Pastramy Project

- Collaborative project involving:
  - INESC-ID
  - University of Minho
  - University of Lisboa

- **Goals:** improve performance and reliability of the FènixEDU system by means of:
  - efficient **transactional memory replication**
  - ad-hoc, lightweight **storage system**

# Our Research Focus

- Our focus is on designing high performance **replication schemes for STM systems**

**Key Observation**

- Databases and TMs share the same fundamental notion of atomic transaction...
- ...database replication schemes represent a natural starting point for STM replication as well!

# An Overview of Database Replication Schemes

# Eager Database Replication

- Classic (eager) database replication relies on:
  - Distributed Two Phase Locking (2PL)
  - Two Phase Commit
- Suffer of large communication overheads:
  - 1 round-trip per data access (to acquire locks)
  - 2 round-trips to commit
- The global serialization order is based on the order of locks' acquisition:
  - High probability of distributed deadlocks as system scales up

# Atomic Broadcast Based DB Replication Schemes (i)

- Rather than relying on distributed locking to tentatively determine a global serialization order, more recent solutions rely on Atomic Broadcast (AB)
- Atomic Broadcast (**<u>key properties</u>**):
  - *If a participant delivers a message, then all correct participants will eventually deliver it*
    **(Uniform Agreement)**
  - *If some participant delivers message A after message B, then every participant delivers B only after it has delivered A*
    **(Uniform Total Order)**

# Atomic Broadcast Based DB Replication Schemes (i)

- Rather than relying on distributed locking to tentatively determine a global serialization order, more recent solutions rely on Atomic Broadcast (AB)
- Atomic Broadcast (**key properties**):
  - *If a participant delivers a message, then all correct participants will eventually deliver it*

    **(Uniform Agreement)**
  - *If some participant delivers message A after message B, then every participant delivers B only after it has delivered A*

    **(Uniform Total Order)**

Whether **correct** or **faulty**: AB encapsulates fault tolerance guarantees

# Atomic Broadcast Based DB Replication Schemes (ii)

## Two main approaches:

### State Machine

1. AB the transaction "code" (e.g. stored procedure)

2. upon AB-delivery: enqueue a lock requests for any data item to be accessed

3. each node runs the transaction only after it acquires all its locks

### Certification

1. optimistically run the transaction on a single node

2. AB the transaction read- and write-set

3. validate the transactions in the order defined by the AB

# AB-based Replication: Pros & Cons

### VS Classic Eager Schemes:

\+ Deadlock-free approaches

\- Single coordination phase

### State Machine

\+ never aborts

\- requires deterministic replicas

\- a-priori knowledge of xacts' read-/write-set

\- all replicas fully execute write transactions

### Certification

\- freq. abort in conflict intensive workloads

\+ not deterministic replicas

\+ no a-priori knowledge of xacts' read-/write-sets

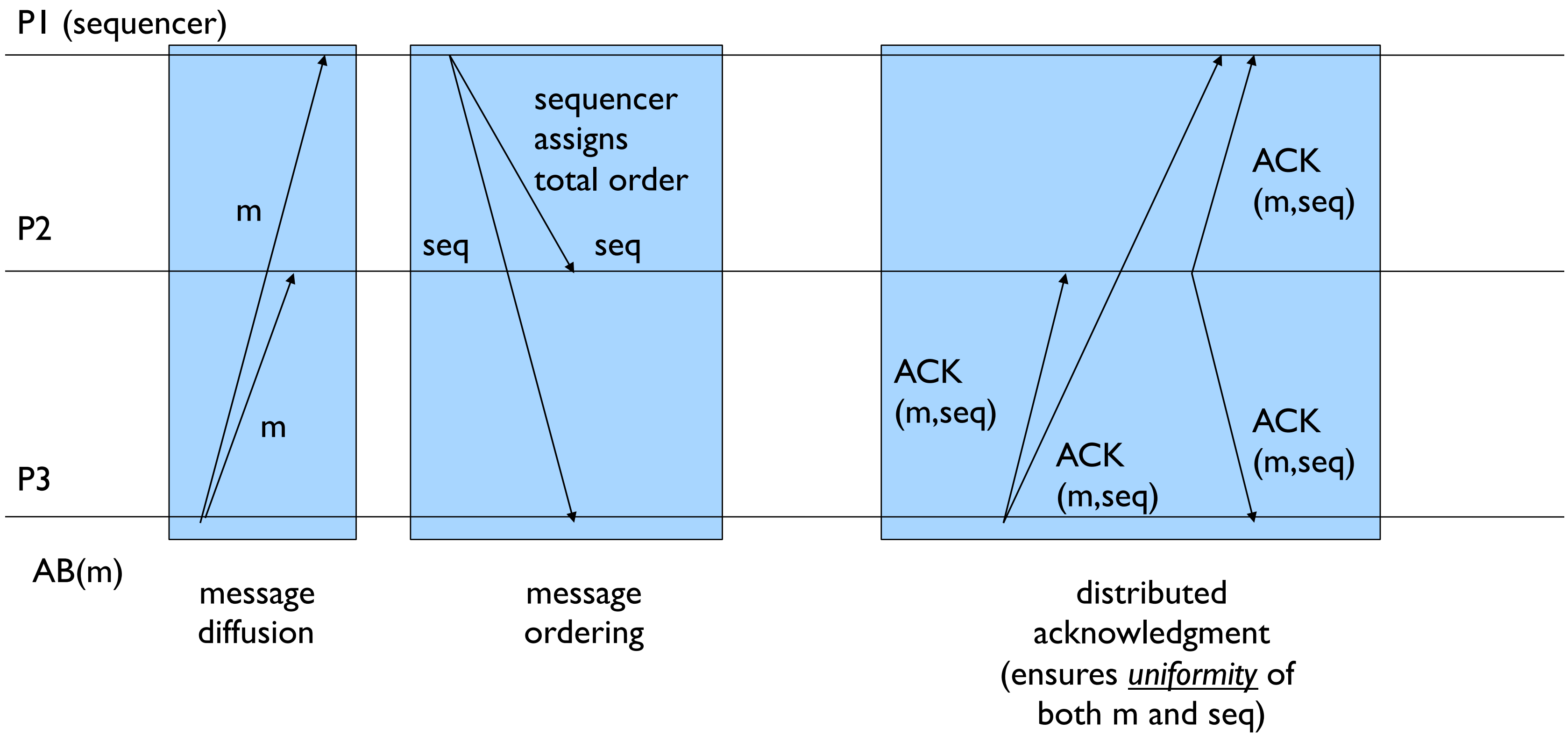
\+ better scale up (potentialities) at high update rates

\- AB is a "relatively" expensive coordination mechanism...

# How expensive is AB in practice?

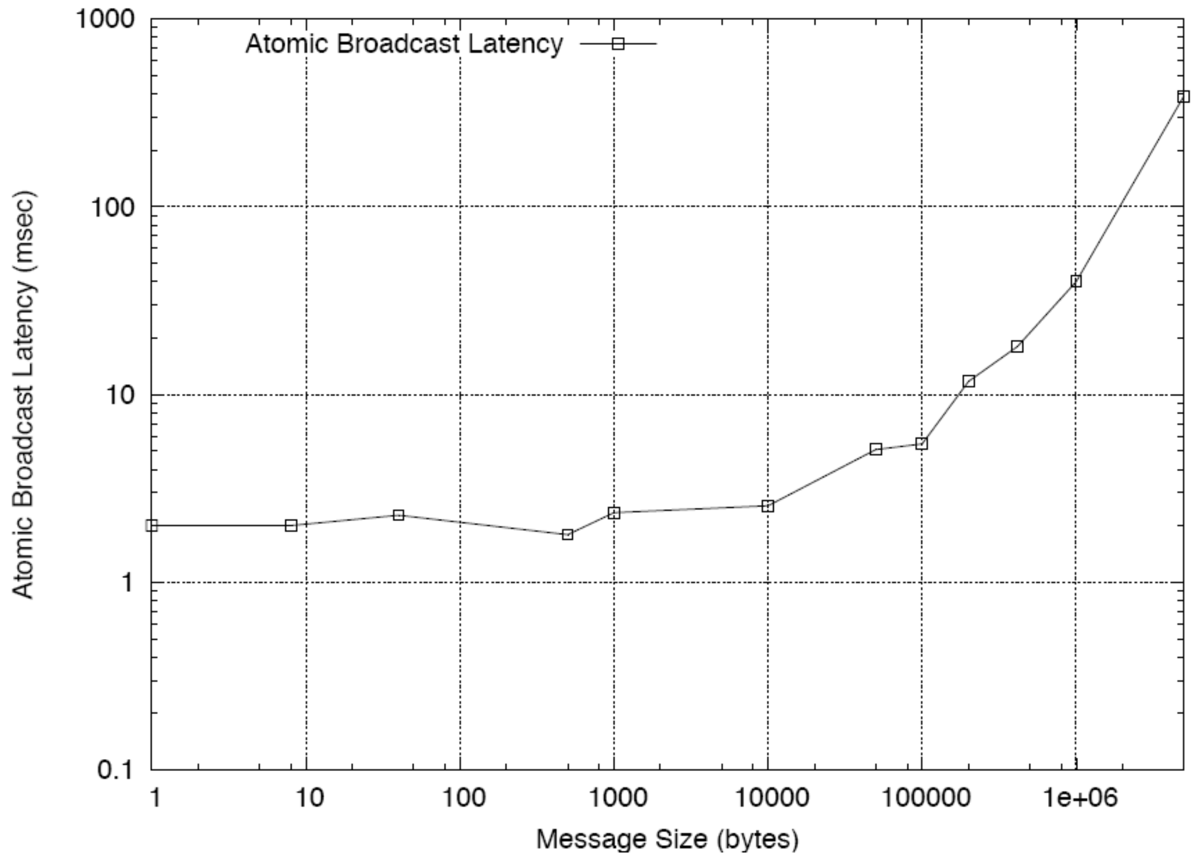
- Hard to give a totally general answer given the huge variety of existing approaches....
- Trade-off between latency and communication complexity:
  - Token-based $\longrightarrow$ Latency: O(n) | Msgs: O(n)
  - Sequencer-based $\longrightarrow$ Latency: 3 | Msgs: $O(n^2)$
- AB performance affected by a number of additional assumptions/ factors, e.g.:
  - Accuracy of failure detection
  - Clock skew among processes
  - Message size

# A simple, low latency AB algorithm




Sequencer based algorithm, 3 nodes, failure-free run

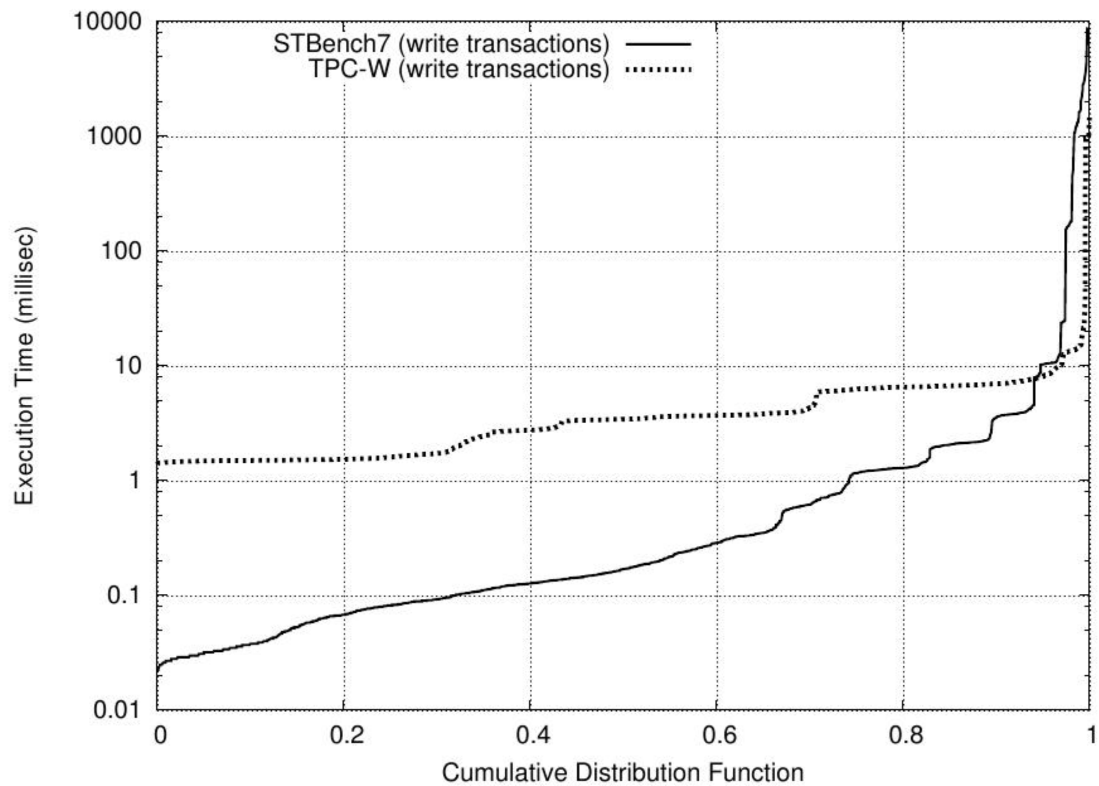
# …and its performances in a LAN




Sequencer based implementation, 3 nodes, FastEthernet, low load, no failures

# Critical Issues for STM Replication

- >70% of xacts are 10-100 times shorter in STMs:
  - correspondingly larger impact of AB latency!
- Transactions' lifetime spans a much wider range in STMs:
  - no "one size fits all" solutions!



# Some of our current research lines

## Certification based approaches

Reduce latency of AB by leveraging the notion of (weak) mutual exclusion

## State machine replication

Overlap coordination and processing phases by executing transactions in speculative serialization orders

# Some of our current research lines

## Certification based approaches

Reduce latency of AB by leveraging the notion of (weak) mutual exclusion

## State machine replication

Overlap coordination and processing phases by executing transactions in speculative serialization orders

# Exploiting mutual exclusion in certification based replication

- Allow nodes to obtain exclusive access to a set of frequently accessed data items
- The "owner" of a set of data items:
  - is sheltered from conflicts with remote transactions:
    - reduce aborts affecting certification based approaches by *bridling* concurrency
  - can play the "equivalent" role of a sequencer in AB:
    - shortcutting message diffusion phase (latency drops to 2 comm. steps)

## **Key Challenges**

- distributed mutual exclusion is solvable only assuming very restrictive synchrony assumptions
- minimize additional overhead for critical section acquisition
- prevent distributed deadlocks

# Fault-Tolerant Distributed Mutual Exclusion Problem

- **Model**
  - $n>1$ processes
  - a *single, indivisible* resource that can only support one process at a time
  - processes can fail by crashing (fail-stop)
  - distributed processes communicating via message passing
- **Problem.** Regulate access to the resource to ensure:

**Safety:** *At any time at most one process is using the resource*

**Liveness:** *Eventually, all correct processes must access the resource*

# Classical Synchrony Models

- Synchronous
  - A priori known bounds on communication latency, relative process speeds and clock drift.
- Asynchronous
  - No bounds exist at all
- Partial Synchrony
  - Bounds exist but are not known in advance
  - Bounds are known, but only hold after some unknown time (Global Stabilization Time, or GST)

# Impossibility Result

- In the presence of faults, the Distributed Mutual Exclusion problem is solvable only in a **synchronous system**

### Why?

Distributed Mutual Exclusion intimately related to the *notion of time*:

**Safety:** *At any time at most one process is using the resource*

### Practical consequences of this theoretical result:

- Real distributed systems are all but synchronous (partitions,overloads)
- Synchronous model can be “approximated” by assuming conservative time-out values, but with :
  - Significant performance drawbacks (large fail-over times)
  - Vulnerability windows can only be statistically minimized, not excluded

# But what if the shared resource is replicated?

- The DME problem considers the case of **<u>fail-prone</u>** distributed processes accessing a **<u>single</u>** **<u>unfailable</u>** resource...
  - ...but in real life processes are not the only ones to fail...
  - ...the shared resource can fail as well !!!

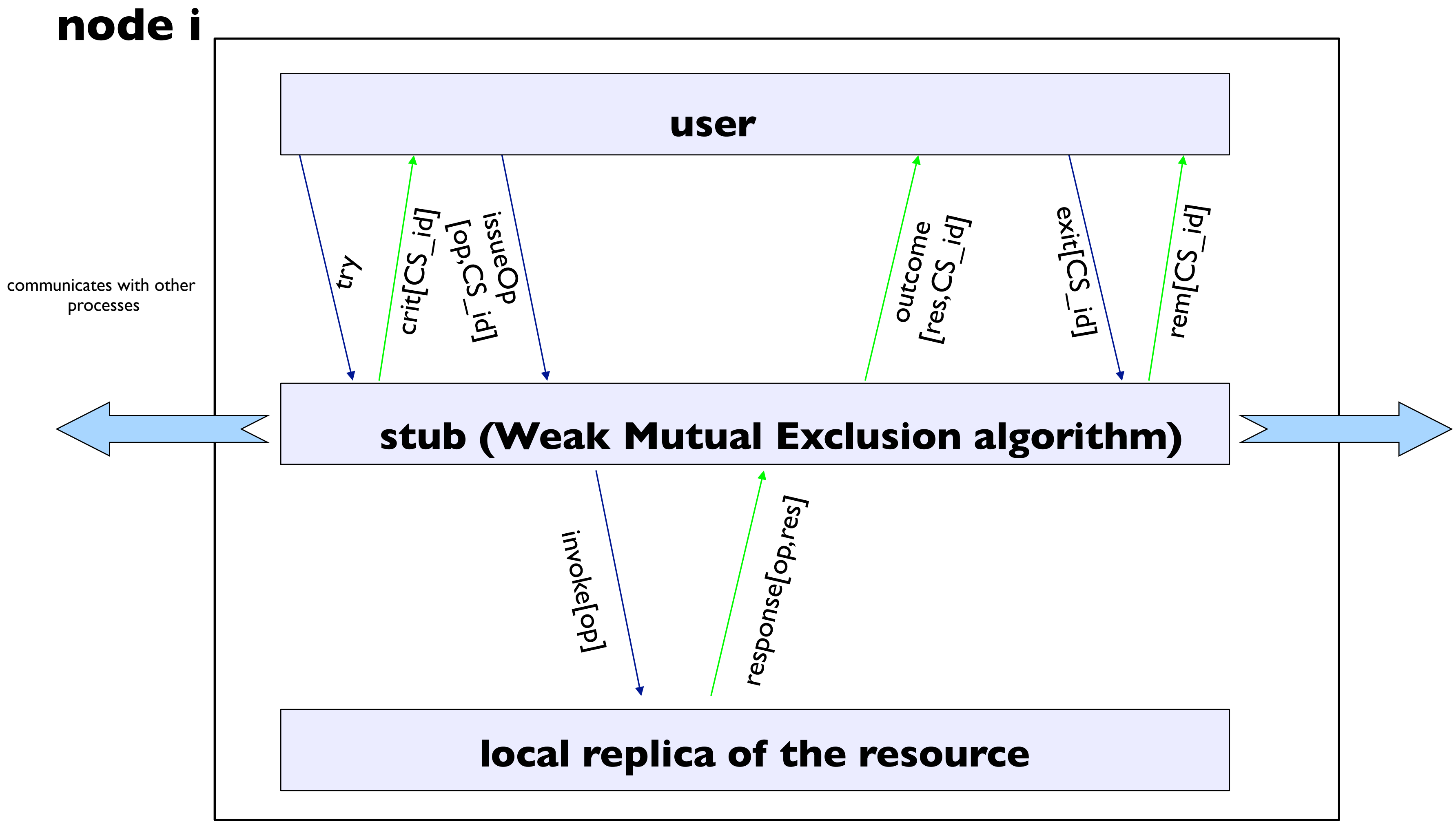
- In real life, e.g. in **<u>STMs</u>**, the shared resource can be:
  - physically replicated across the processes
  - required to appear as a single resource accessible in mutual exclusion

# The Weak Mutual Exclusion Problem

- Is the problem of mutual exclusion in the access to a replicated shared resource as hard as classical ME?
  - is it solvable in a more relaxed model?
  - what is the minimum level of synchrony needed?
  - ...but first of all how to formally define it?

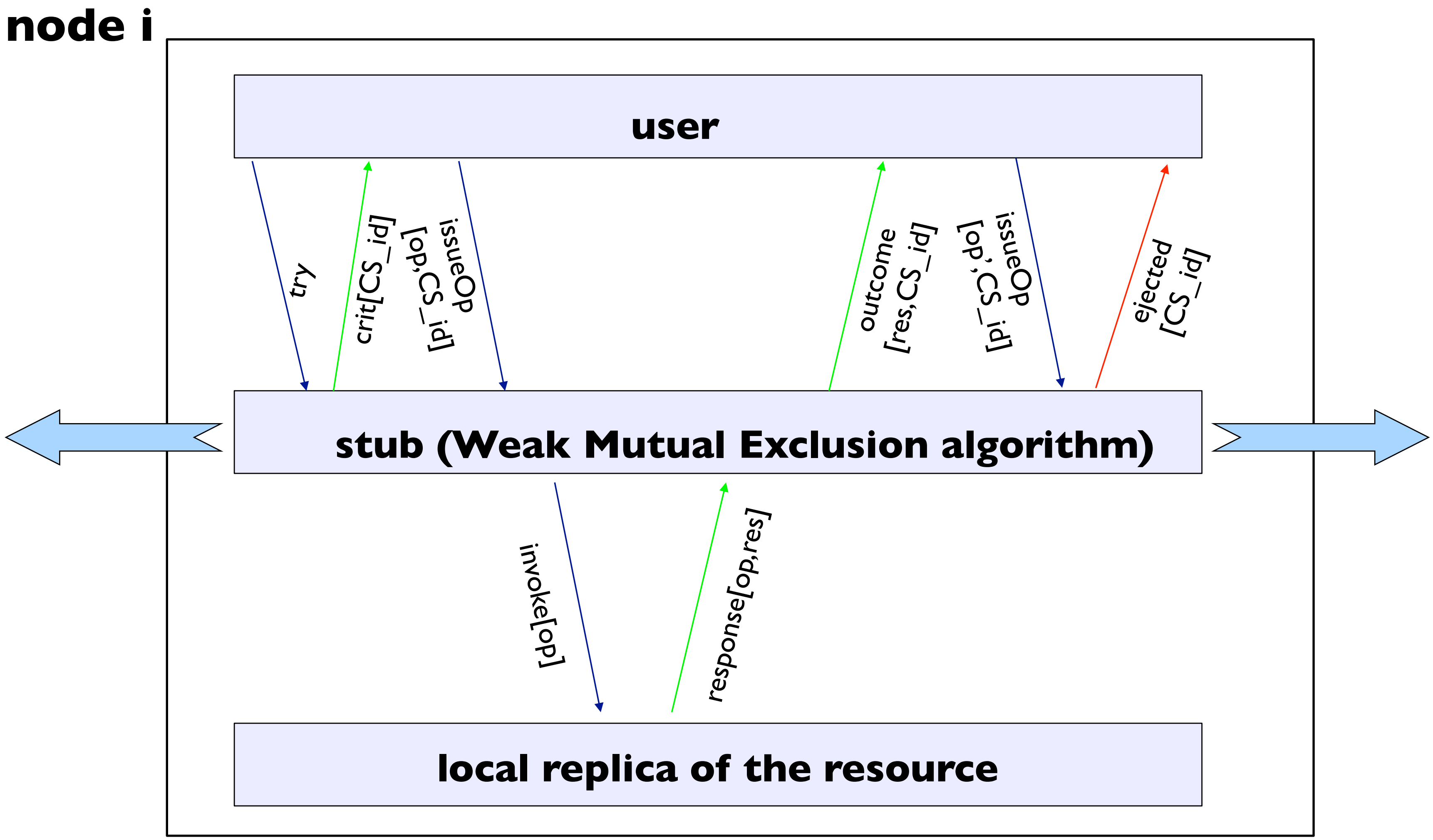
- 3 key ingredients differentiating from classic DME:
  1. Explicitly **modelling the interactions with the resource**
  2. Bound to the notion of **logical time**, **rather than global time**
  3. Admit the possibility of being **ejected by the critical section**

# System Model - I




**Well behaved run – no ejection from mutual exclusion**

# System Model - II




**Run incurring an <u>ejection</u> from mutual exclusion**

# The Weak Mutual Exclusion Problem Specification: Safety Properties

**Weak Mutual Exclusion (informal definition)**

It is possible to reorder the global execution history so to:

- yield an equivalent sequential history, S, in which all successful operations appear executed in a sequence of critical sections (CS)
- preserve the local order of events at each single process
- ensure that the order of acquisition of the CSs in S is consistent with the order of acquisition of not overlapping CSs in the original history

**One Copy Serializability**

S is equivalent to a sequential execution on a not replicated resource

**Well-formedness**.

Rules out malformed interactions, e.g. duplicate mutual exclusion request...

# The Weak Mutual Exclusion Problem Specification: Liveness Properties

**Starvation-Freedom**.

A correct process wishing to enter the CS, eventually enters the CS, if no other process stays forever in its CS

**CS-Release Progress.**

No correct process blocks when releasing the CS

**Operation Progress.**

If a correct process issues an operation, it eventually gets an outcome, and eventually all the operations it issues succeed.

# The Weak Mutual Exclusion Problem Solvability Results

- **<u>Good news:</u>**
  - We proved that the WME problem is solvable in a partially synchronous system model...
  - ...and precisely quantified the minimum necessary degree of synchrony based on Chandra-Tueg's failure detection hierarchy
- <u>So, how to exploit WME in a replicated STM?</u>

# WME-based Certification

1) Partition data into conflict classes

2) Process update xacts at a single replica

3) At commit time, acquire locks (unless not already owned) on the accessed conflict classes:

- AB lock request, piggybacking transaction read-/write-sets:
  - Acquiring lock based on AB's total order guarantees deadlock-freedom and consistent evolution despite failure (suspicions)
  - No additional messages for lock acquisition

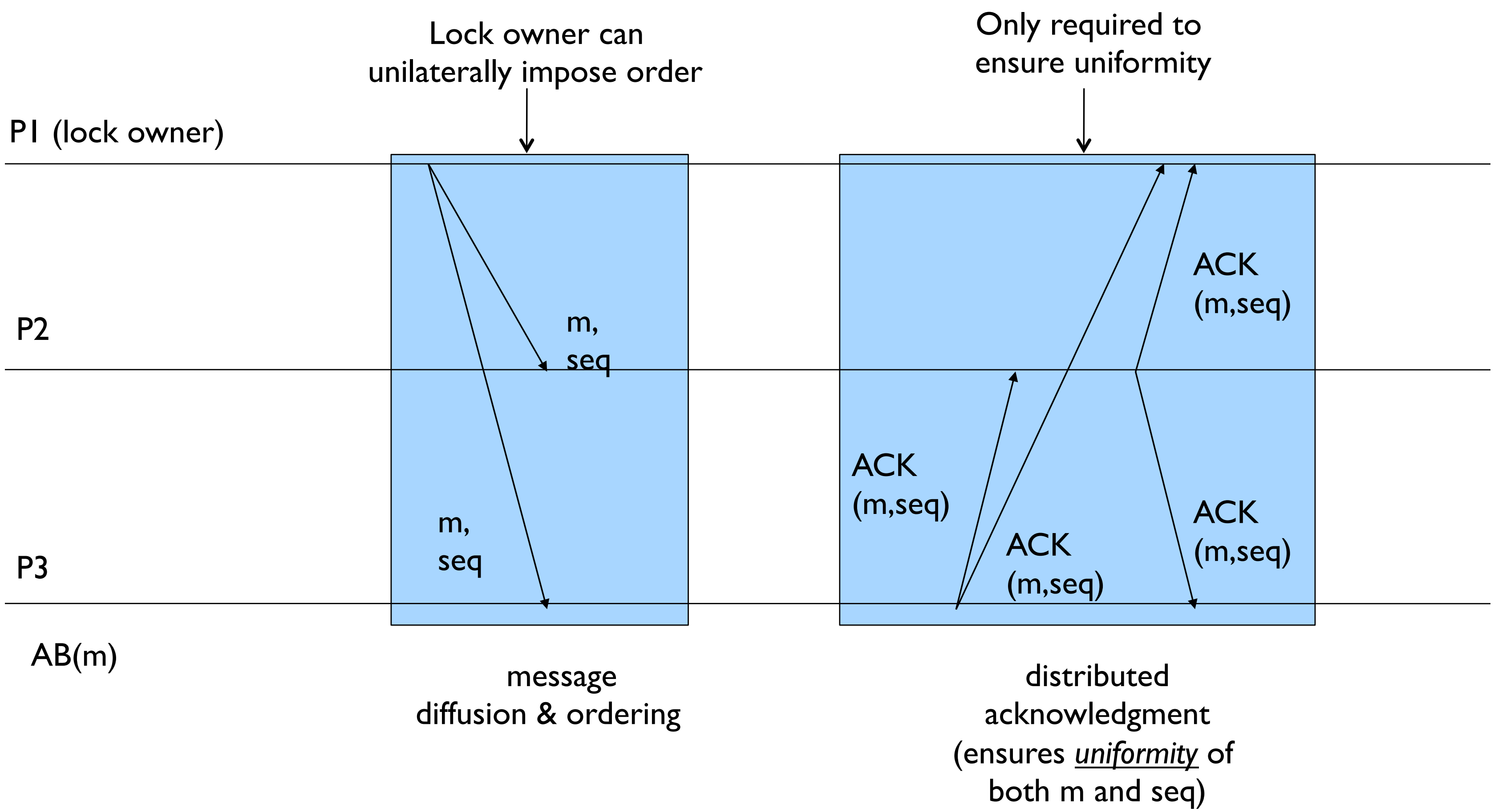
If the locks are locally already granted:

- Lock owner can globally impose its local xact serialization order

  *(1)* ***avoiding remote conflicts***

  *(2) saving* ***one communication step*** *wrt* ***AB***

# WME-based Certification




Locked data is guaranteed to be accessed in mutual exclusion (unless the lock owner is ejected due to a failure suspicion)

# Some of our current research lines

### Certification based approaches

Reduce latency of AB by leveraging the notion of (weak) mutual exclusion

### State machine replication

Overlap coordination and processing phases by executing transactions in speculative serialization orders

# Problem

WME-based certification scheme works well:

- If there is some data locality:
  - or lock ownership may start to be “ping-ponged”
- If the conflict probability is not very high:
  - absence of remote conflicts only after the lock is acquired

- In these scenarios it may be worth to rely on an abort-free <u>state machine replication</u> approach

# Key Idea

Overcome two main problems of state machine replication approach:

- A-priori knowledge of transaction's read-/write-sets
- Reduce the impact of AB's latency on system's performance

**How?**

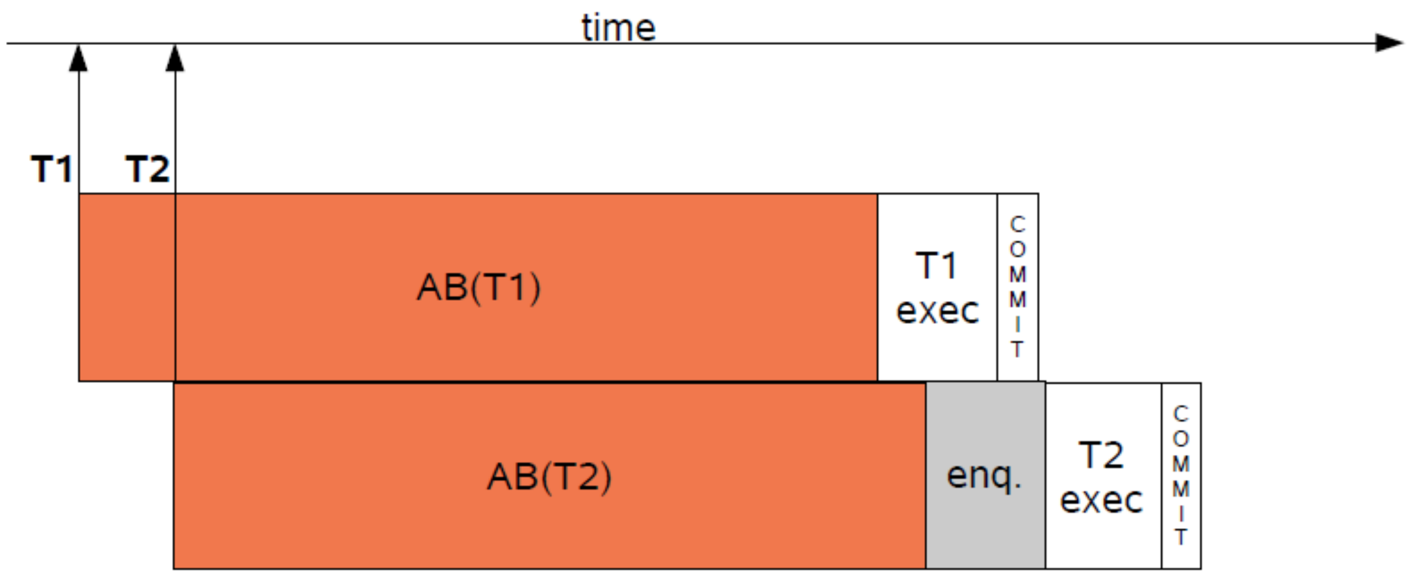
- Exploiting atomic broadcast with optimistic delivery
- Using extra CPU cycles to explore alternative serialization orders

# Optimistic Atomic Broadcast

Atomic broadcast with **two** delivery indications:

- **Optimistic delivery.** Early estimate of the final total order:
  - Typically available after a single communication step
  - Possibly contradicted by final delivery
  - e.g., local network receive order
- **Final delivery.** Agreed (uniform) total order

# State Machine Replication

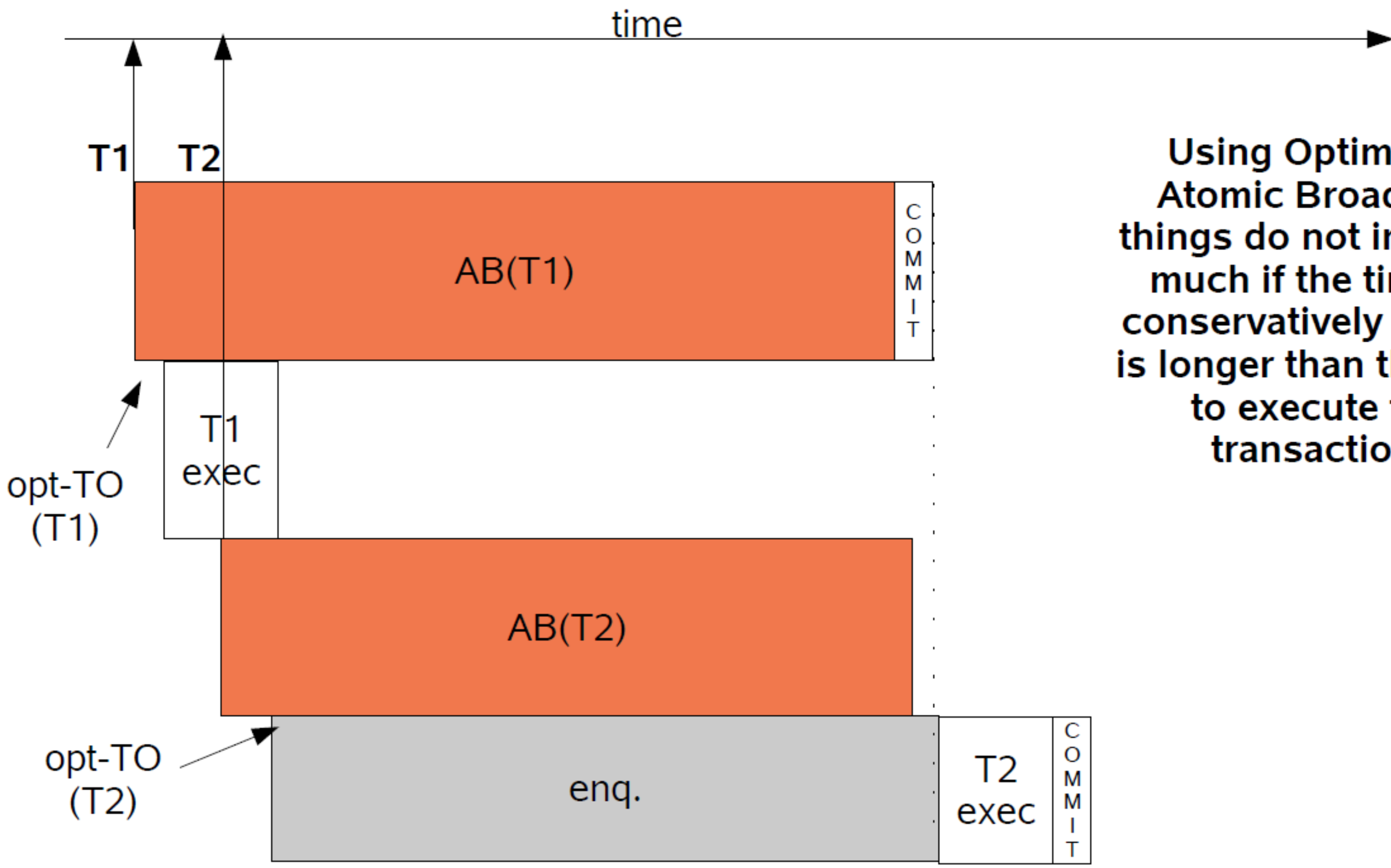



• A priori knowledge of read-/write-sets required to ensure deterministic scheduling of concurrent conflicting transactions:

- hard to **exactly** determine a priori, **normally coarsely over-estimated**...

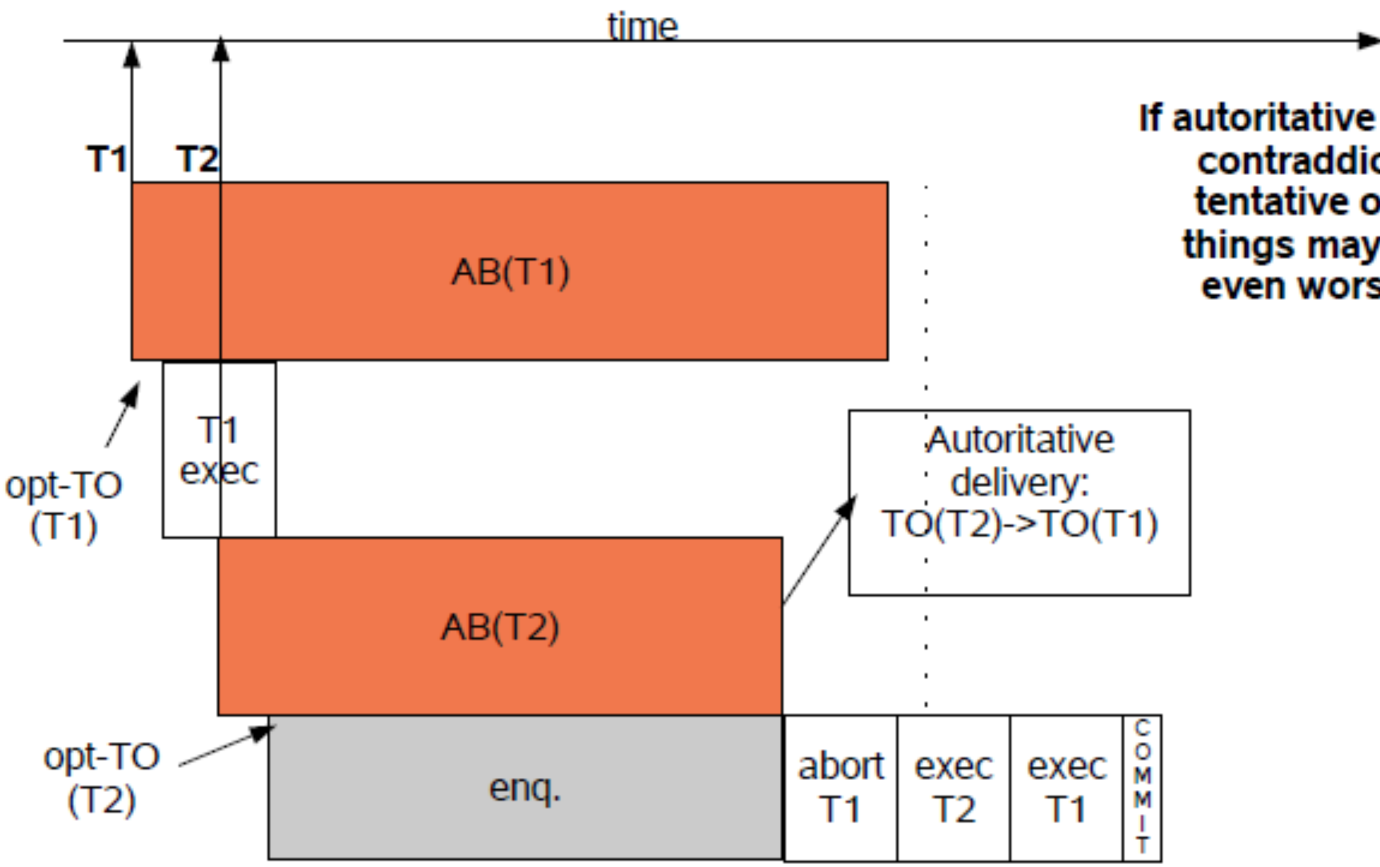
• Relatively large AB latency may cause severe resource underutilization

# Non-speculative use of Optimistic Atomic Broadcast



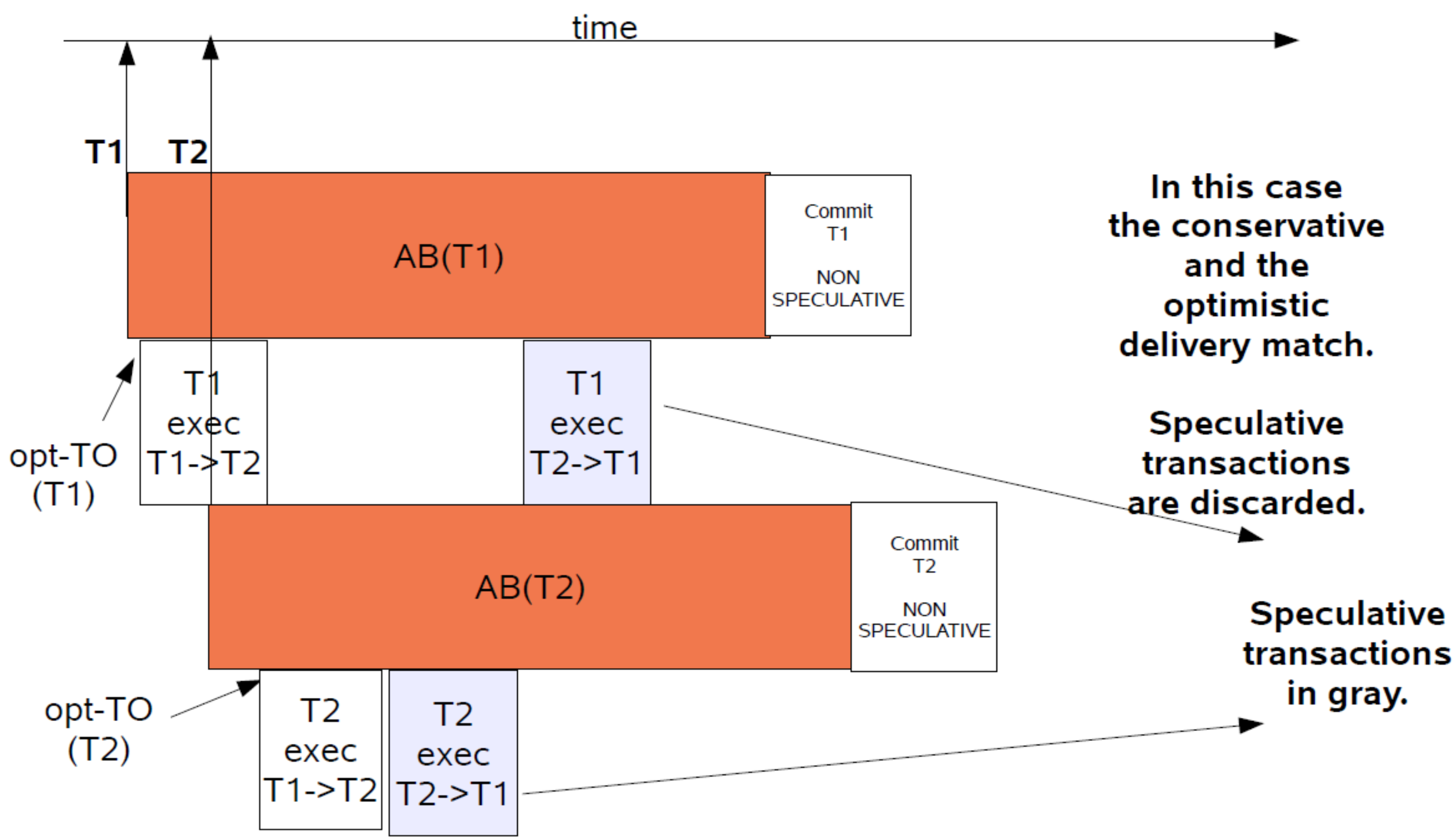

**Using Optimistic Atomic Broadcast things do not improve much if the time to conservatively deliver is longer than the time to execute the transaction.**

# Optimistic delivery giving erroneous indications...




**If autoritative order contraddicts tentative one things may go even worse!**

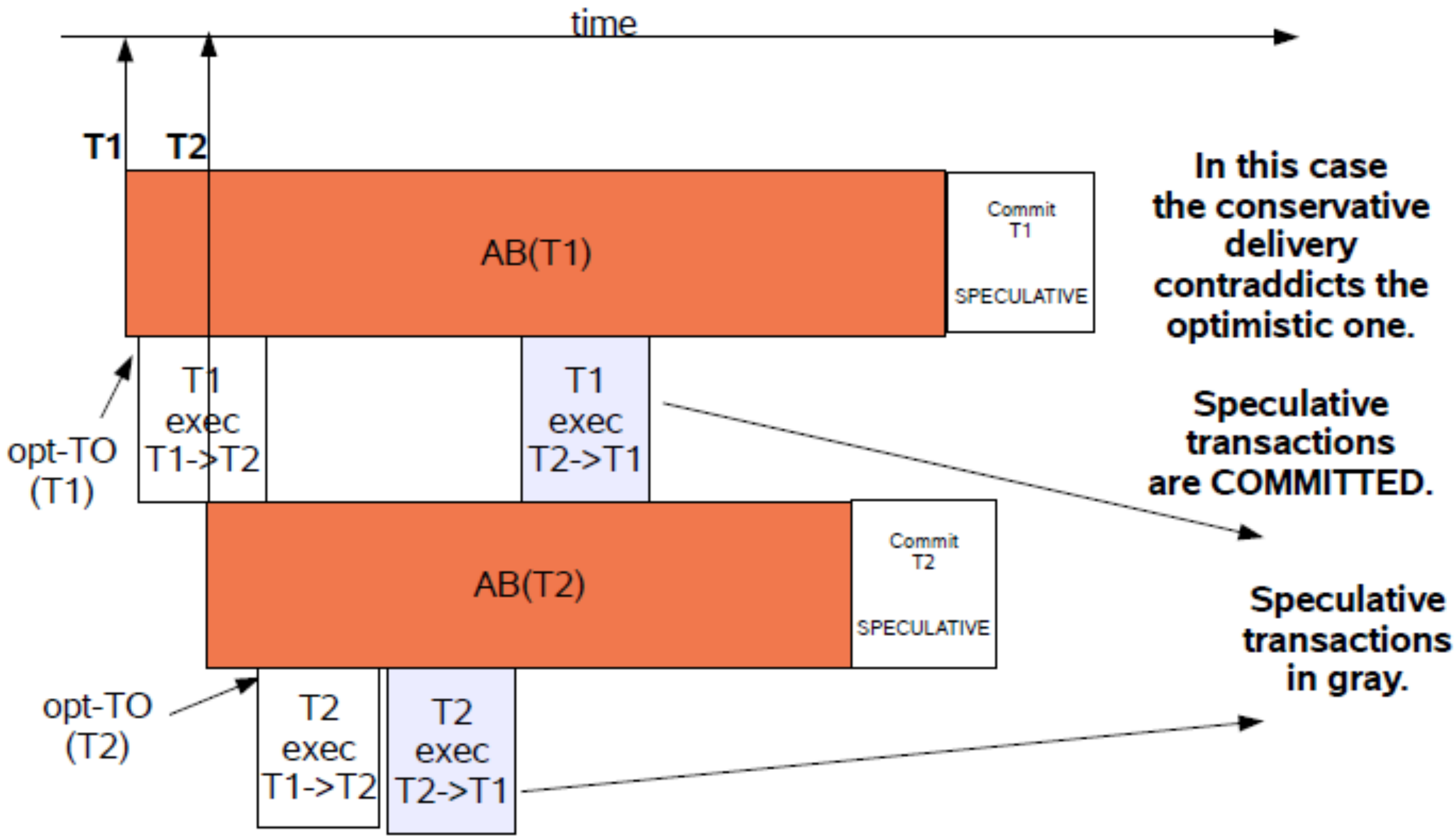
# Speculative Transaction Processing




The need for exploring alternative speculative serialization orders is determined depending on the actual conflicts generated by the execution of the opt-delivered transactions:

- Without any a-priori knowledge on their read-/write-sets

# Speculative Transaction Processing



# Challenges

1. Determine the set of alternative serialization orders (s.o.) to be explored based on run-time determined transactions' conflict relations:
   - note that a transaction's read-/write-sets may change if this is re-executed in a different serialization order

2. Define a concurrency control scheme allowing transactions to observe a speculative database snapshot representative of a given *s.o.*

3. Introduce effective heuristics to identify which ones among the O(N!) *s.o.*s to be actully activated based on:
   - the likelihood of the various possible permutations
   - the current availability of idle computing resources

# Other Research Directions

# Space efficiency via Bloom Filters

**Problem:**

- most efficient AB-based replication schemes require exchanging transactions readsets...
- ...which can be huge, drastically affecting the AB latency!

**Idea:** exploit Bloom Filters space efficient encoding to deterministically limit the message size

**Challenge:** (efficiently) accommodating unavoidable false positives

# Self-adapting Replication Strategies

**Problem:** No single replication protocol is able to optimally cope with the high heterogeneity of STM based systems

**Idea**: Develop STM replication protocols able to self-adapt depending on, e.g.:

- transaction's object-set size
- estimated transaction conflict probability
- ...

**Challenge:** allow consistent coexistence of multiple replication schemes

# Conclusions

- Distributed STMs may provide a robust, scalable and fault-tolerant solution for building applications (including web-based ones)
- Distributed STMs can be built using ideas from the distributed shared memory and database replication areas
- New research problems appear when one attempts to combine both worlds. Some example:
  - Weak Mutual Exclusion
  - Speculative Transaction Processing

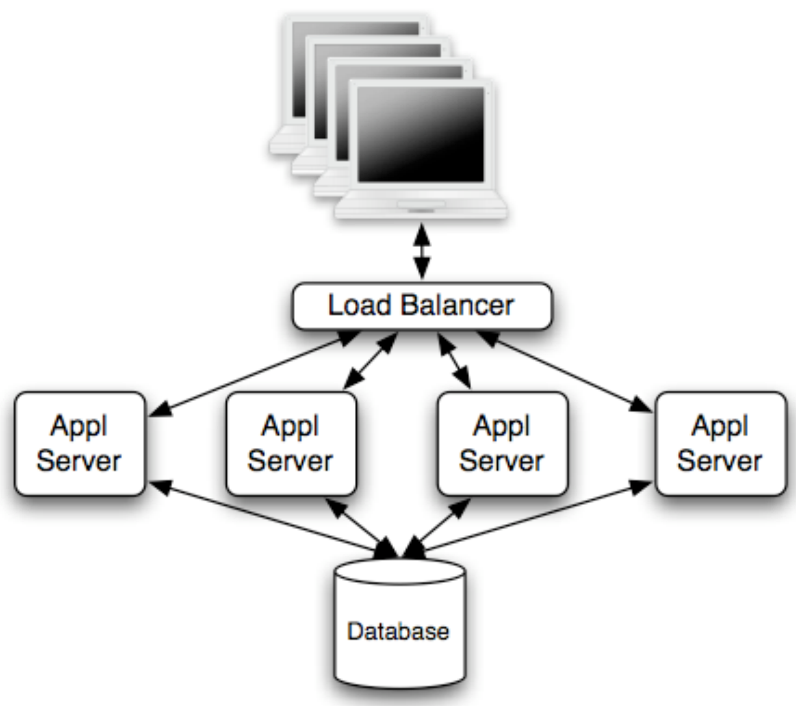
Thanks for
the attention

The FenixEDU system is
freely available at the following URL:

**https://fenix-ashes.ist.utl.pt/**

# Backup
# Slides

# Current FénixEDU Architecure




- Replicated application servers
- Replica synchronization through centralized validation at the back-end database

**Open Problems**

- Interface with relational DBMS consumes an excessive amount of memory
- DBMS is the system's bottleneck and single point of failure

# Expected Future Architecture