STM Contention Management: from Multiprocessors to Distributed Systems

Danny Hendler
Ben-Gurion university

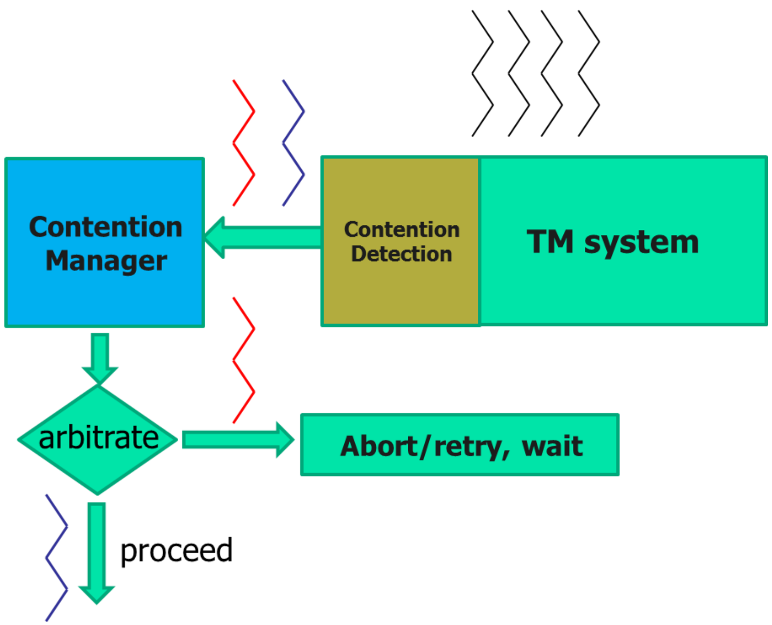
What is TM contention management?

"

When two or more transactions attempt to access the same block of transactional memory concurrently, at least one transaction must be aborted. The decision of which transaction to abort, and under what conditions, is the **contention management problem**

"

[Scherer and Scott, contention management in dynamic STM, DISC'04]



"STM for dynamic-sized data structures", Herlihy, Luchangco, Moir & Scherer PODC'03

What is TM contention management?

"

When two or more transactions attempt to access the same block of transactional memory concurrently, at least one transaction must be aborted. The decision of which transaction to abort, and under what conditions, is the **contention management problem**

"

[Scherer and Scott, contention management in dynamic STM, DISC'04]

"

The mechanisms used [byTM implementations] to ensure forward progress – to avoid livelock and starvation, and to promote throughput and fairness.

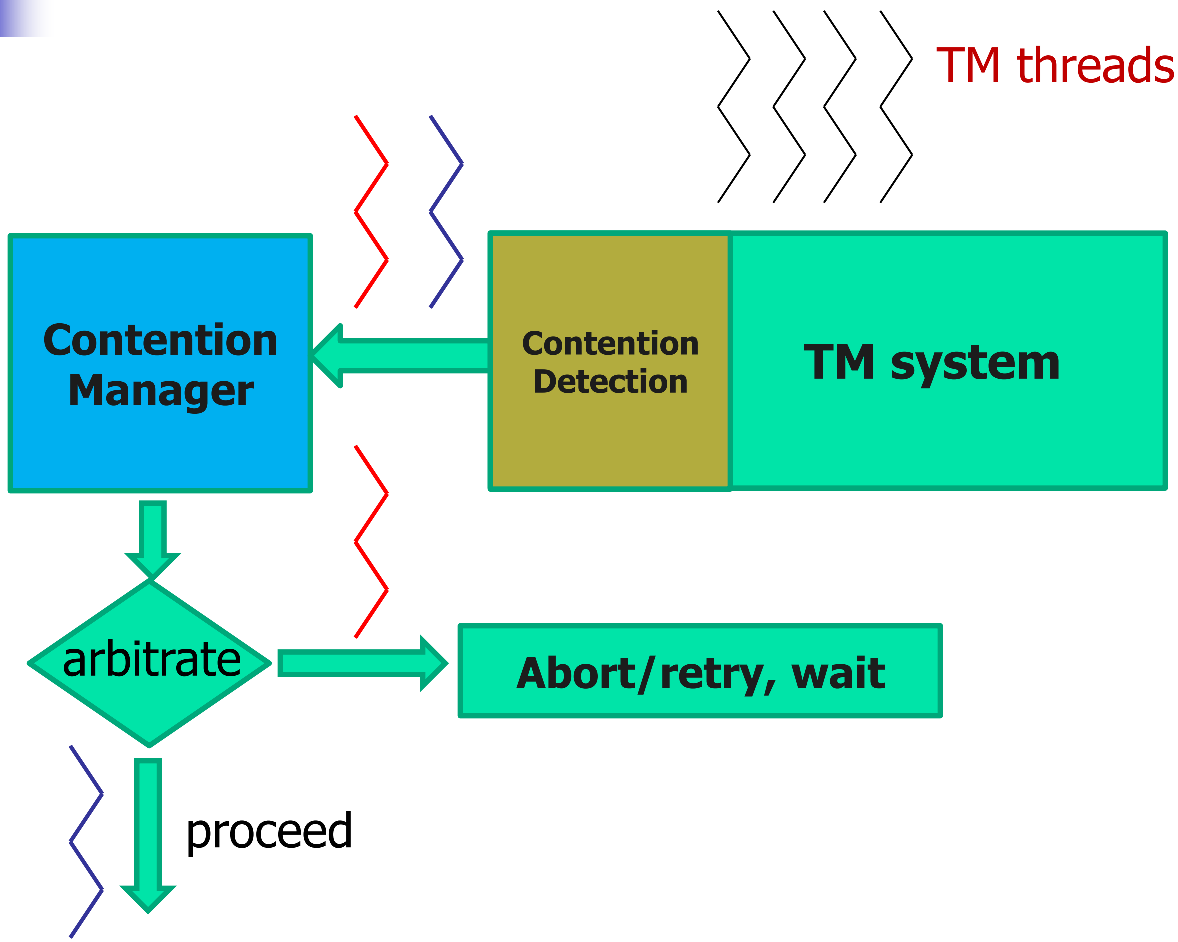
"

[Spear, Dalesandro, Marathe & Scott, A comprehensive strategy for contention management in STM, PPoPP'09]

Talk outline

- Preliminaries
- Multiprocessor contention management
  - Conflict resolution policies
  - TM schedulers
- Distributed TM (DTM)
  - Design space and principles
  - Example replicated DTM implementations
  - Contention management considerations

"Conventional" conflict resolution policies



Passive
Aggressive
Polite
Karma
greedy
Polka

"STM for dynamic-sized data structures", [Herlihy, Luchangco, Moir & Scherer PODC'03]

"Polymorphic contention management", [Guerraoui, Herlihy & Pochon DISC'05]

Conventional conflict resolution policies are often insufficient

- Loser resumes execution after pre-determined waiting period
  - May resume execution too early
  - May resume execution too late
- Repeated collisions occur under high contention
  - Livelocks
  - Performance may become worse than single lock

Scheduling-based CM to the rescue.

Talk outline

- Preliminaries
- Multiprocessor contention management
  - Conflict resolution policies
  - TM schedulers
- Distributed TM (DTM)
  - Design space and principles
  - Example replicated DTM implementations
  - Contention management considerations

TM schedulers: rationale

- Transactional threads controlled by TM-aware scheduler
  - Kernel-level, user-level

- Richer “tool-box“ for reducing and/or preventing transaction conflicts

⇩

Improve performance under high-contention

The first TM schedulers

- “Adaptive Transaction Scheduling for transactional memory systems”, Yoo & Lee, SPAA'08

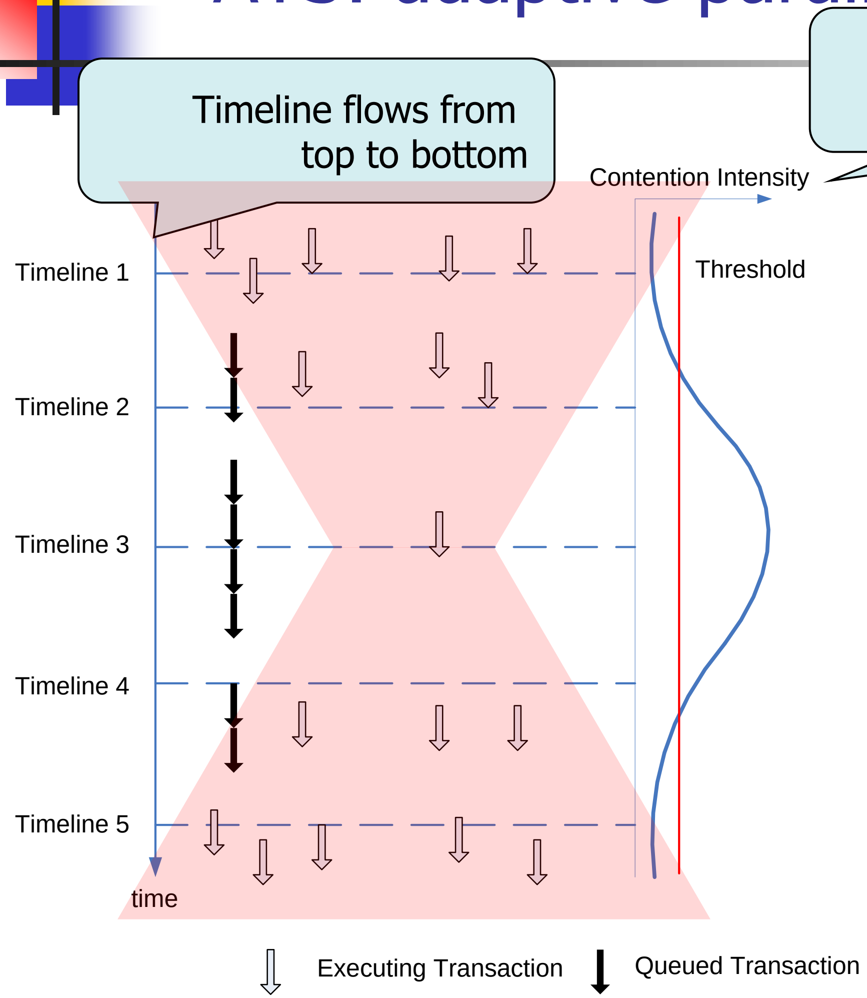
- “CAR-STM: Scheduling-based collision avoidance and resolution for software transactional memory”, Dolev, Hendler & Suissa, PODC '08

- “Steal-on-abort: dynamic transaction reordering to reduce conflicts in transactional memory”, Ansari , Jarvis, Kirkham, Kotsedilis, Lujan and Watson, HiPEAC'09

Adaptive Transaction Scheduling (ATS) Yoo & Lee, SPAA'08

- A single scheduling queue
- Per-thread <u>Contention Intensity</u> (CI) computed
- Adaptive mechanism
  - CI below threshold ➔ transaction begins normally
  - CI above threshold ➔ transaction serialized (queued)

ATS: adaptive parallelism control



Behavior of a Queue-Based Scheduler

Transactions begin execution without resorting to the scheduler

As contention starts to increase, some transactions call the scheduler

As more transactions get serialized, contention intensity starts to decrease

Contention intensity subsides below threshold

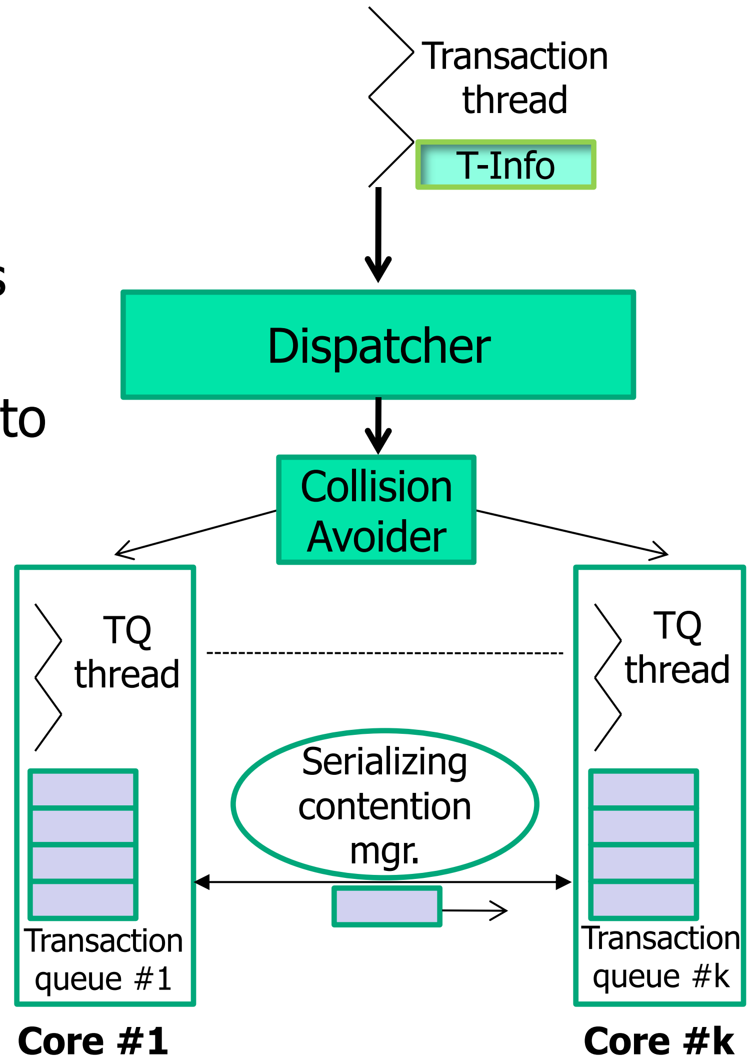
More transactions start without the scheduler to exploit more parallelism

ATS adaptively varies the number of concurrent transactions according to the dynamic contention feedback

Yoo & Lee, Transaction Scheduling.

CAR-STM: Collision Avoidance and Resolution for STM
Dolev, Hendler & Suissa, PODC'08

- ❑ Per-core transaction queues
- ❑ Serialize conflicting transactions
- ❑ Contention avoidance: attempt to avoid even first collision



Additional TM scheduling work

- “Abort Free SemanticTM by Dependency Aware Scheduling of Transactional Instructions”, Dolev, Fatourou & Kosmas, WTTM'13
- “Transaction scheduling using dynamic conflict avoidance” Nicácio, , Baldassin & Araújo, IJPP'13
- “On the impact of serializing contention management on STM performance” Heber, Hendler & Suissa, JPDC'12
- “Transactional scheduling for read-dominated workloads” Attiya & Milani, JPDC’12
- “Window-based greedy contention management for TM” Sharma, Estrade & Busch, DC'12
- “Kernel-assisted Scheduling and Deadline Support for STM” Maldonado, Marlier, Felber, Lawall, Muller & Riviere, DSN'11
- “Adaptive thread scheduling techniques for improving scalability of STM” Chan, Lam & Wang, 2011
- “Scheduling support for transactional memory contention management” Maldonado, Felber, Suissa, Hendler, Fedorova, Lawall & Muller, PPoPP'10
- “Improving performance by reducing aborts in HTM” Ansari, Khan, Lujan, Kotselidis, Kirkham and Watson, HIPEAC'10
- …

Talk outline

- Preliminaries
- Multiprocessor contention management
  - Conflict resolution policies
  - TM schedulers
- Distributed TM (DTM)
  - Design space and principles
  - Example replicated DTM implementations
  - Contention management considerations

Distributed Transactional Memory (DTM)

- Extends the reach of TM abstraction to distributed applications
- Enhanced scalability, high-availability and fault-tolerance
- Large design space

DTM design choices: memory abstraction

- ❑ Uniform global address space
  - ✓ Simple programming model
  - ✓ Same API as multiprocessor TM
  - x No programmatic control of data/code locality

- ❑ Partitioned Global Address Space (PGAS)
  - ✓ Explicit distinction between local and remote partitions allows optimizing for locality
  - x Complicates programming model

DTM design choices: memory abstraction

- ❑ Uniform global address space
  - ✓ Simple programming model
  - ✓ Same API as multiprocessor TM
  - x No programmatic control of data/code locality

- ❑ Partitioned Global Address Space (PGAS)
  - ✓ Explicit distinction between local and remote partitions allows optimizing for locality
  - x Complicates programming model

DTM design choices: memory abstraction

- ❑ Uniform global address space
  - ✓ Simple programming model
  - ✓ Same API as multiprocessor TM
  - x No programmatic control of data/code locality

- ❑ Partitioned Global Address Space (PGAS)
  - ✓ Explicit distinction between local and remote partitions allows optimizing for locality
  - x Complicates programming model

DTM design choices: execution model

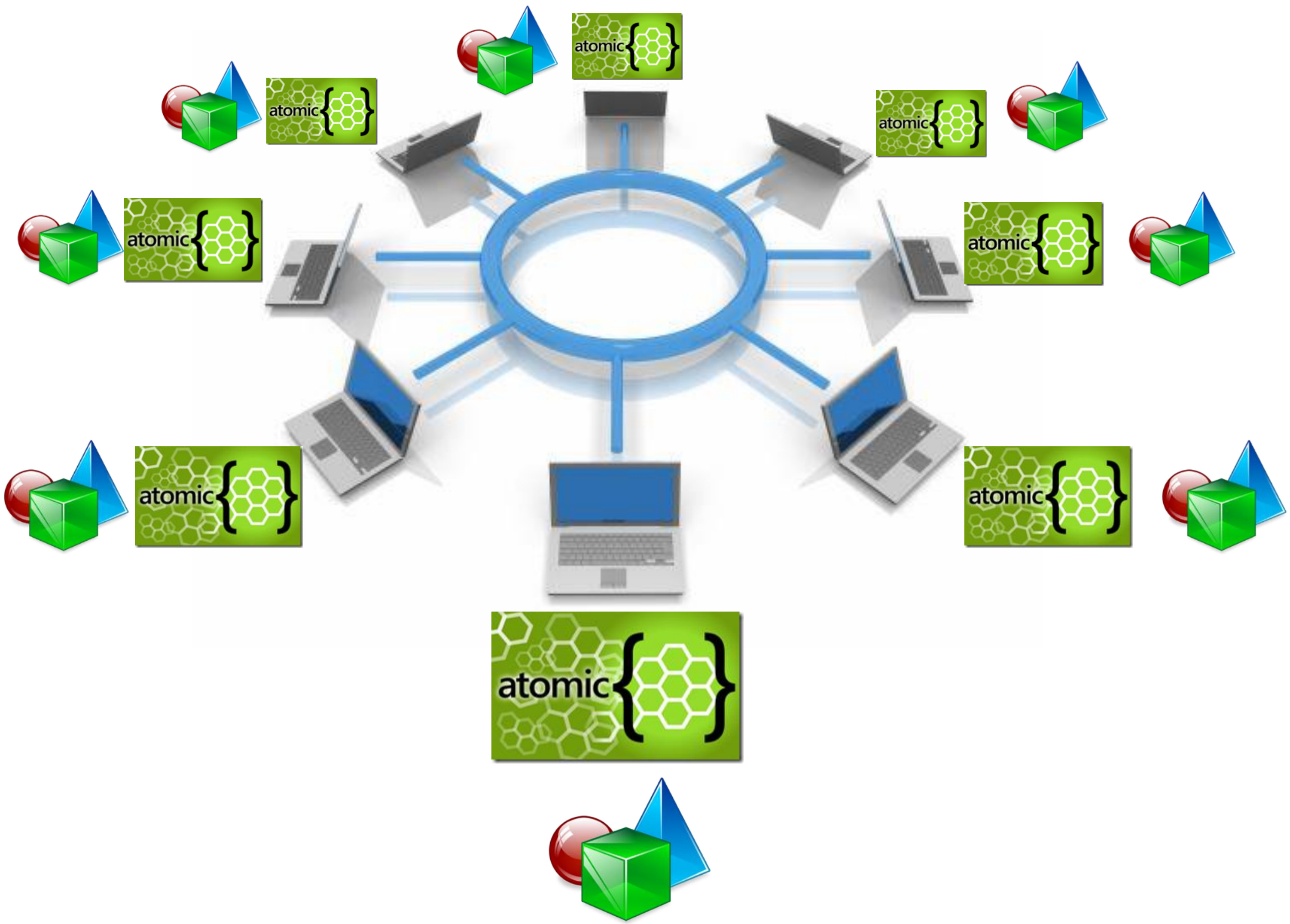
- ❑ Control flow (objects statically assigned to home nodes)
  - ✓ Fast data location
  - ✓ Easy integration of caching schemes
  - x Poor data locality
- ❑ Data flow (transactional code immobile)
  - ✓ Potentially good data locality
  - ✓ Local validation & commit possible
  - x Locating and mobilizing data may be costly

DTM design choices: execution model

- ❑ Control flow (objects statically assigned to home nodes)
  - ✓ Fast data location
  - ✓ Easy integration of caching schemes
  - x Poor data locality

- ❑ Data flow (transactional code immobile)
  - ✓ Potentially good data locality
  - ✓ Local validation & commit possible
  - x Locating and mobilizing data may be costly

DTM design choices: replication

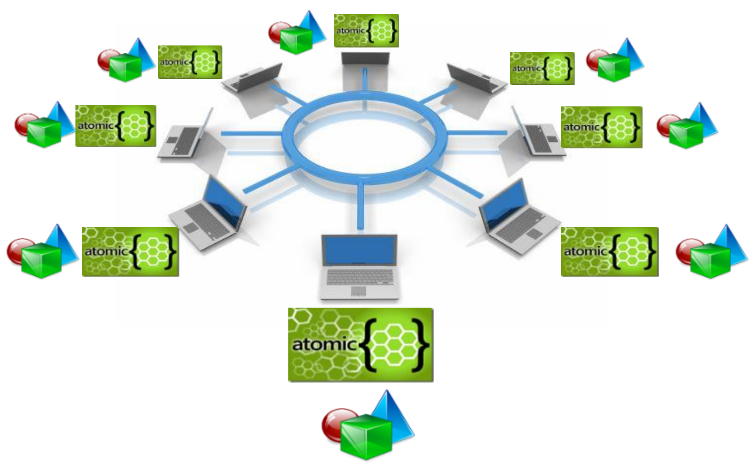


Replicated DTM: advantages & challenges

√ Fault-tolerance

√ Read-only transactions may avoid remote communication

x Communication required for maintaining data consistency

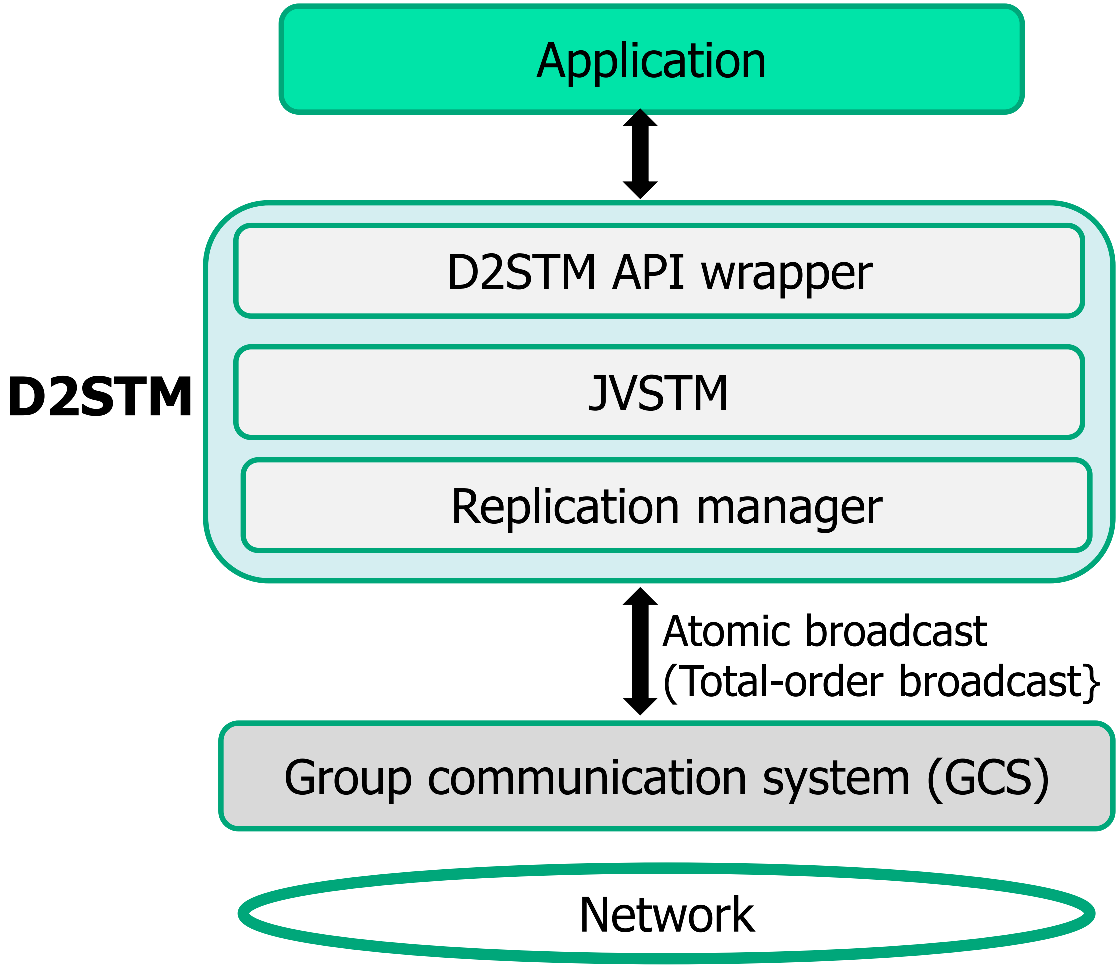


Talk outline

- Preliminaries
- Multiprocessor contention management
  - Conflict resolution policies
  - TM schedulers
- Distributed TM (DTM)
  - Design space and principles
  - Example replicated DTM implementations
  - Contention management considerations

Dependable Distributed STM (D2STM)

Coucerio, Romano, Carvalho & Rodrigues, PRDC'09

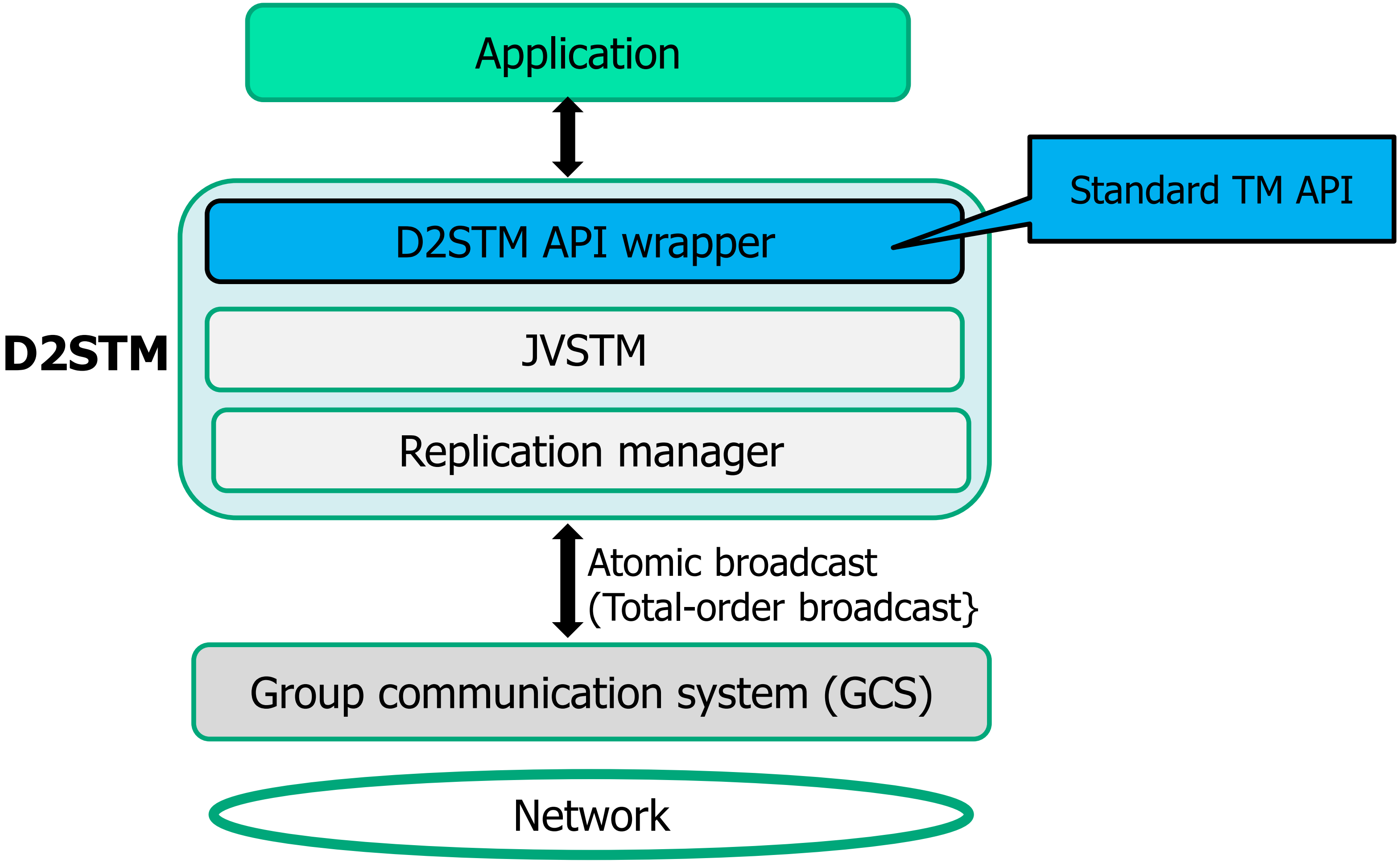


"Tutorial on Distributed Transactional Memories", Romano & Rodrigues

"Versioned boxes as the basis for memory transactions", Cachopo & Silva, SCP'06

Dependable Distributed STM (D2STM)

Coucerio, Romano & Carvalho, PRDC'09

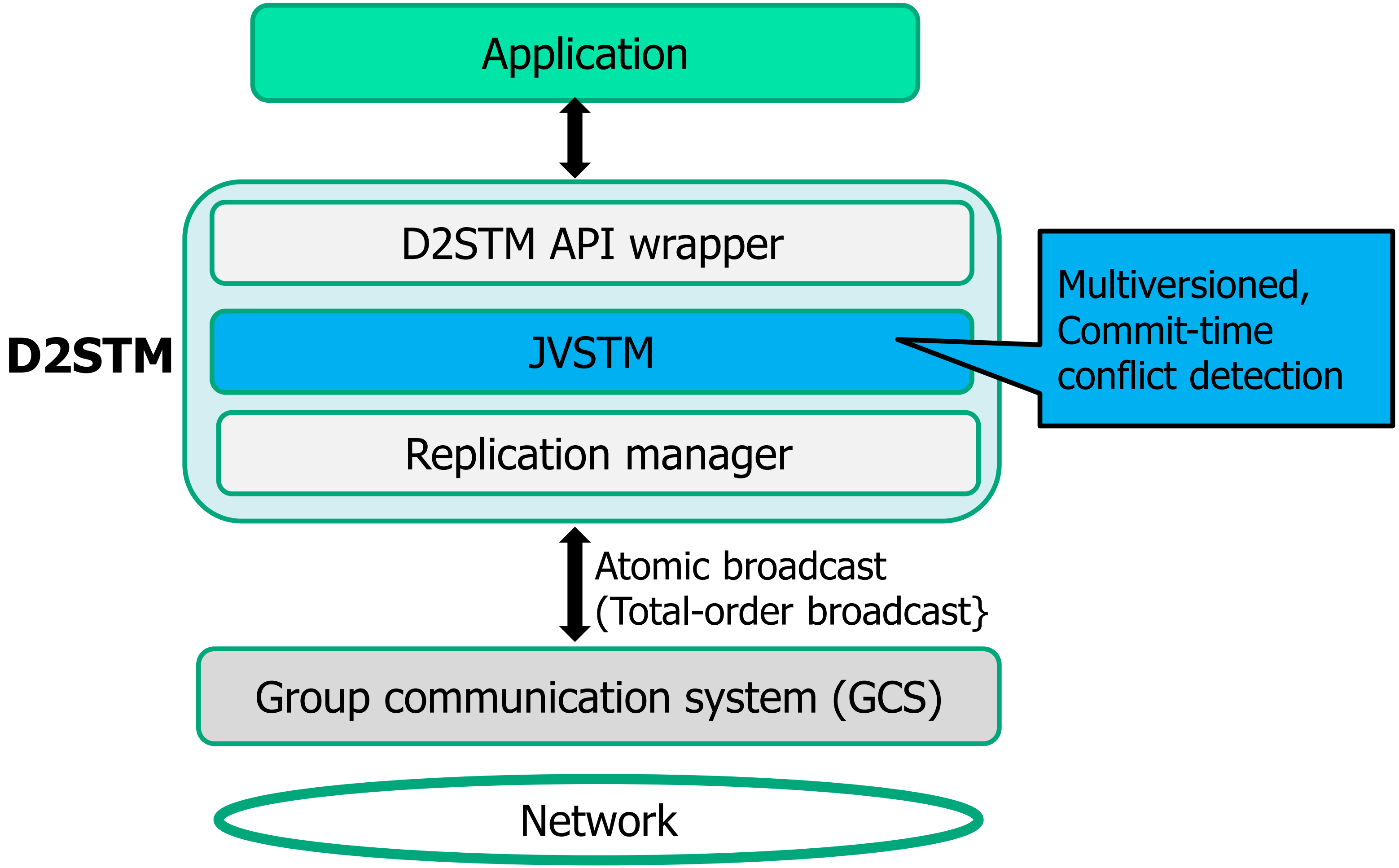


"Tutorial on Distributed Transactional Memories", Romano & Rodrigues

"Versioned boxes as the basis for memory transactions", Cachopo & Silva, SCP'06

Dependable Distributed STM (D2STM)

Coucerio, Romano & Carvalho, PRDC'09

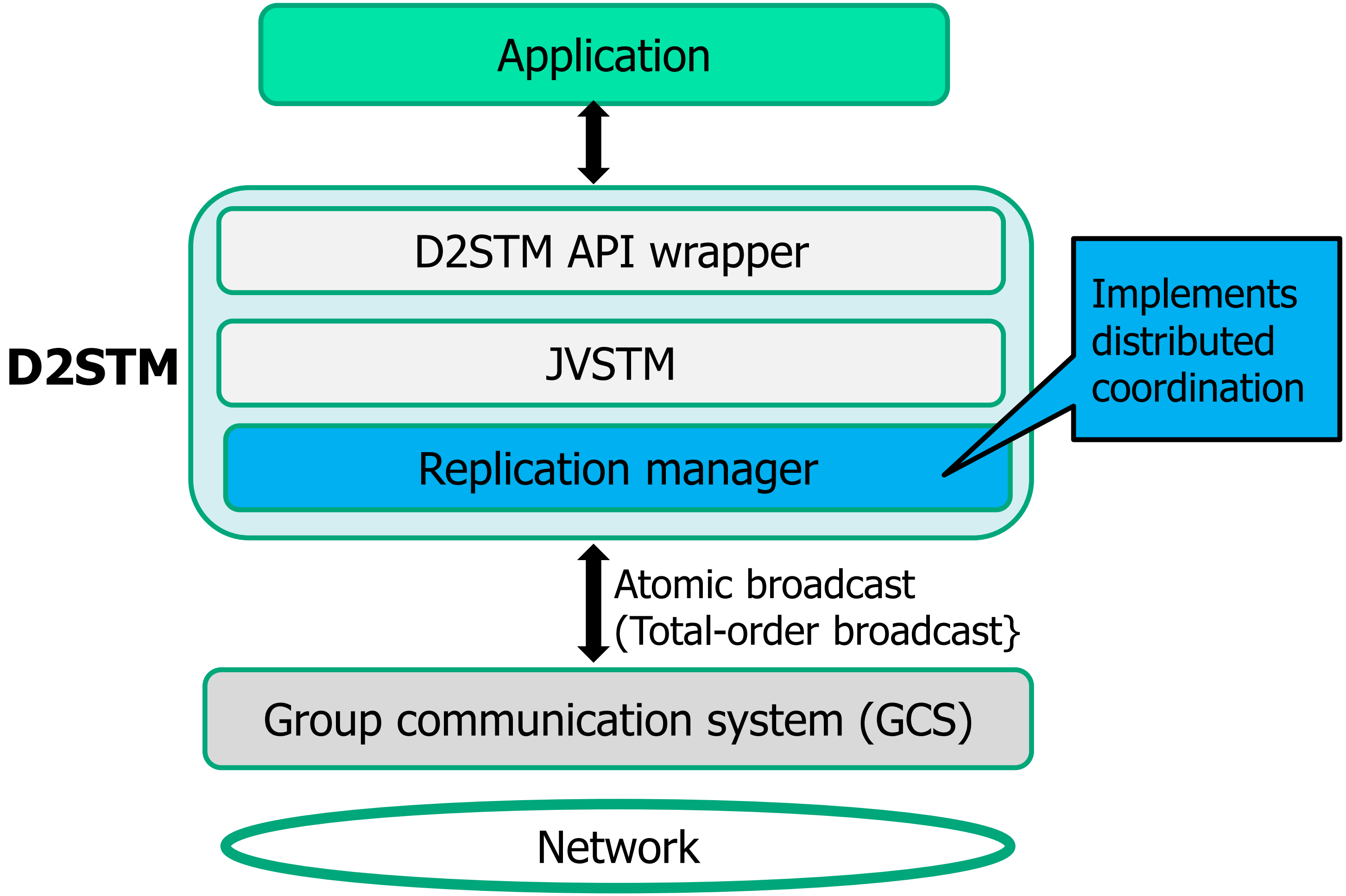


"Tutorial on Distributed Transactional Memories", Romano & Rodrigues

"Versioned boxes as the basis for memory transactions", Cachopo & Silva, SCP'06

Dependable Distributed STM (D2STM)

Coucerio, Romano & Carvalho, PRDC'09

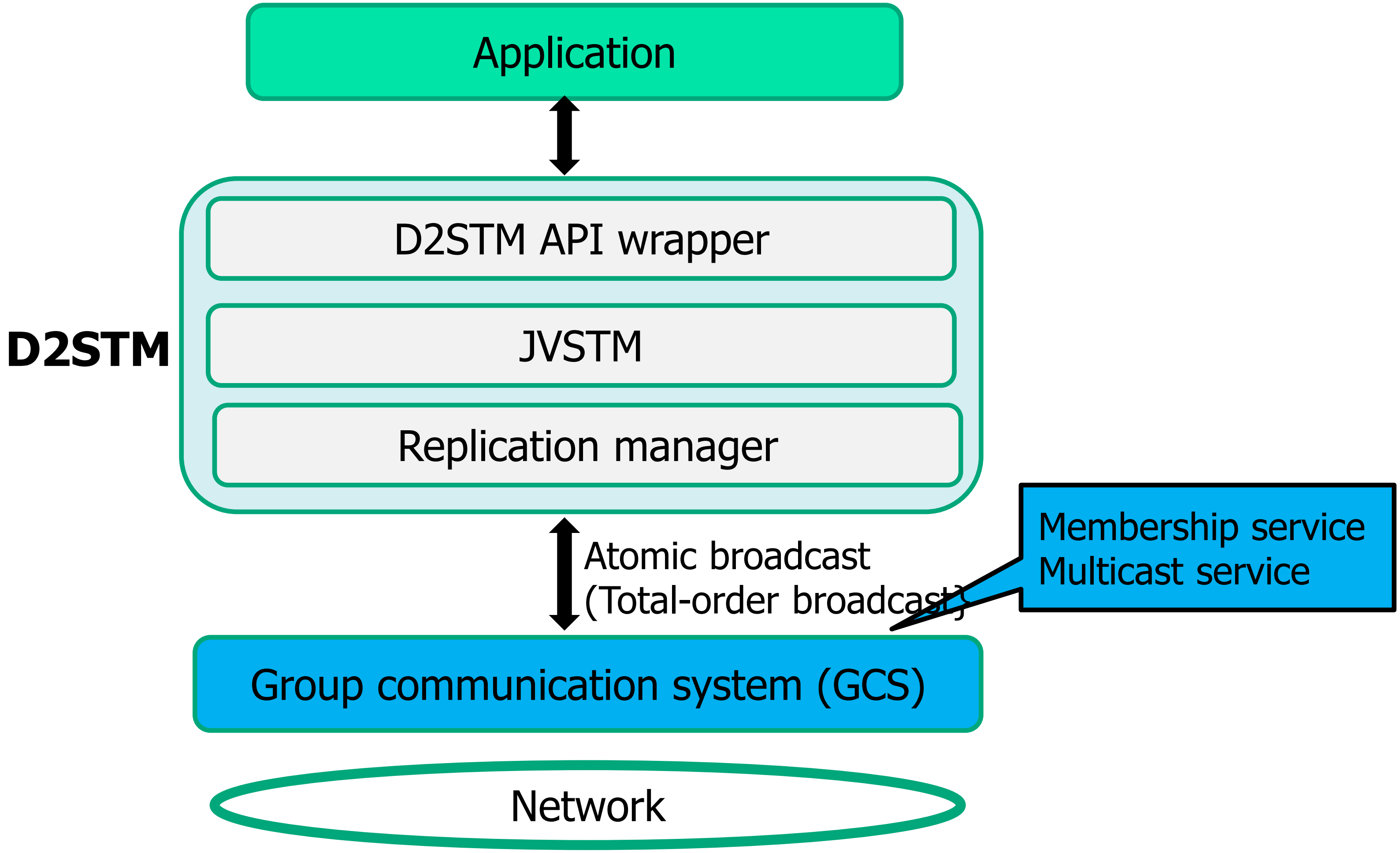


"Tutorial on Distributed Transactional Memories", Romano & Rodrigues

"Versioned boxes as the basis for memory transactions", Cachopo & Silva, SCP'06

Dependable Distributed STM (D2STM)

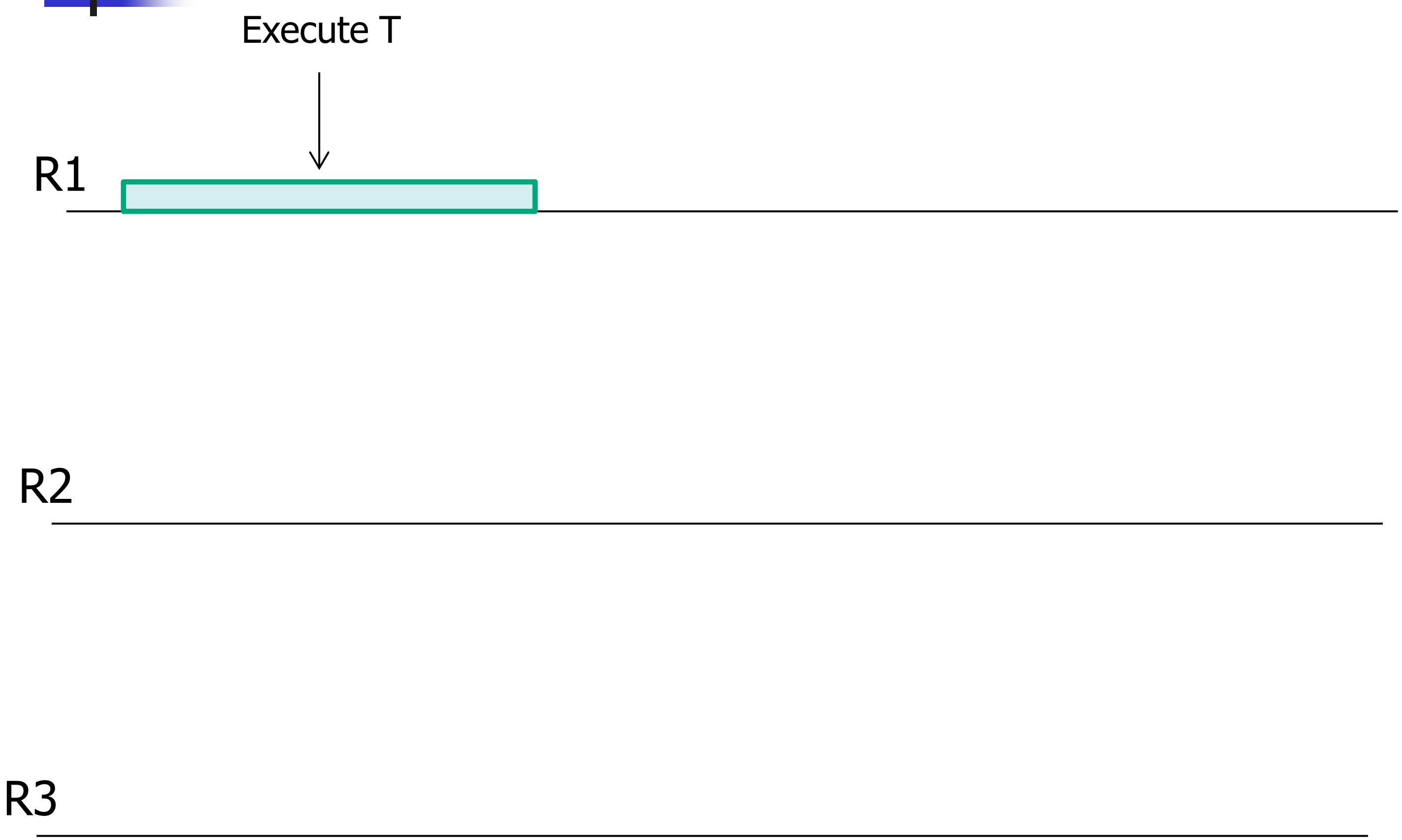
Coucerio, Romano & Carvalho, PRDC'09



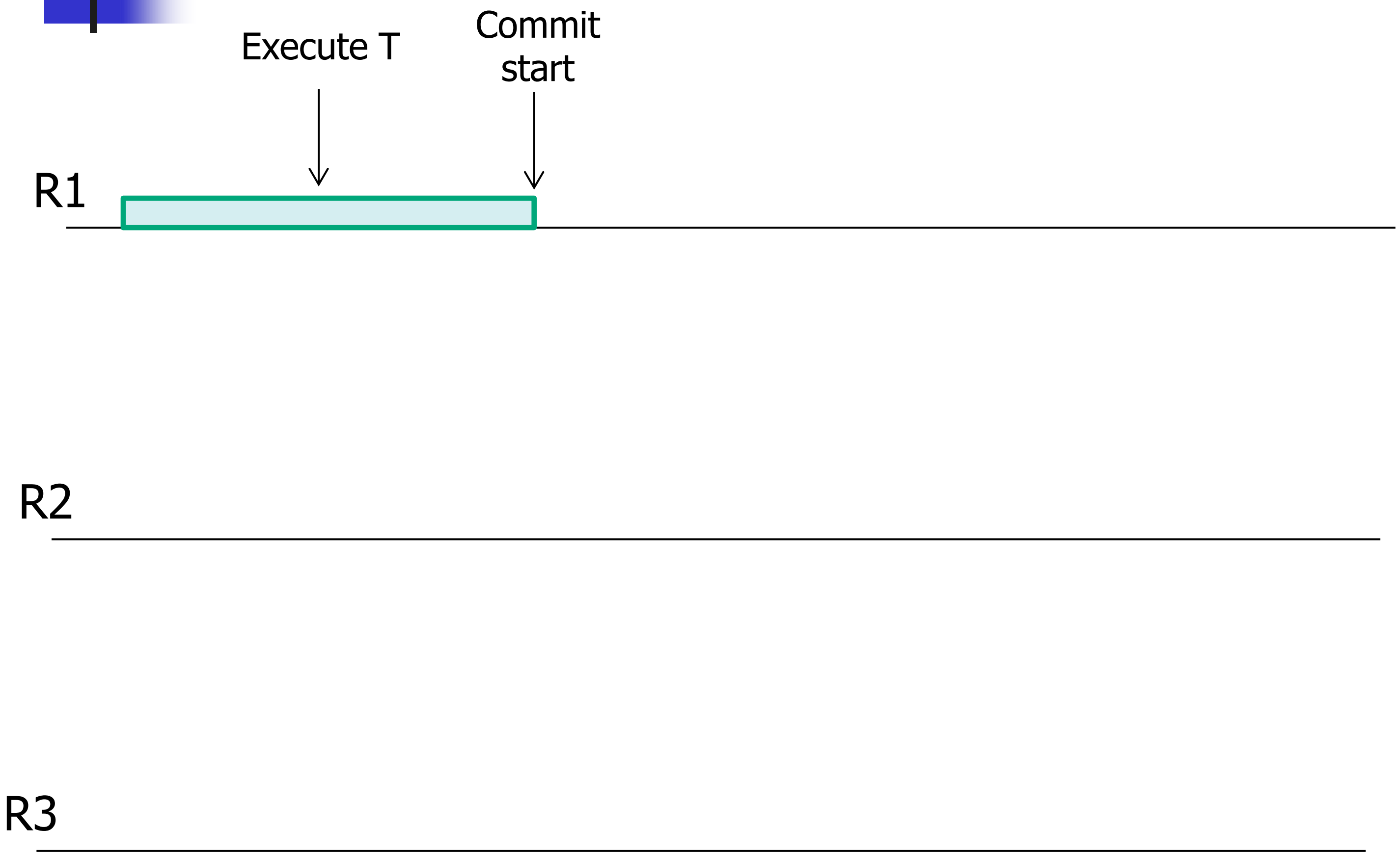
"Tutorial on Distributed Transactional Memories", Romano & Rodrigues

"Versioned boxes as the basis for memory transactions", Cachopo & Silva, SCP'06

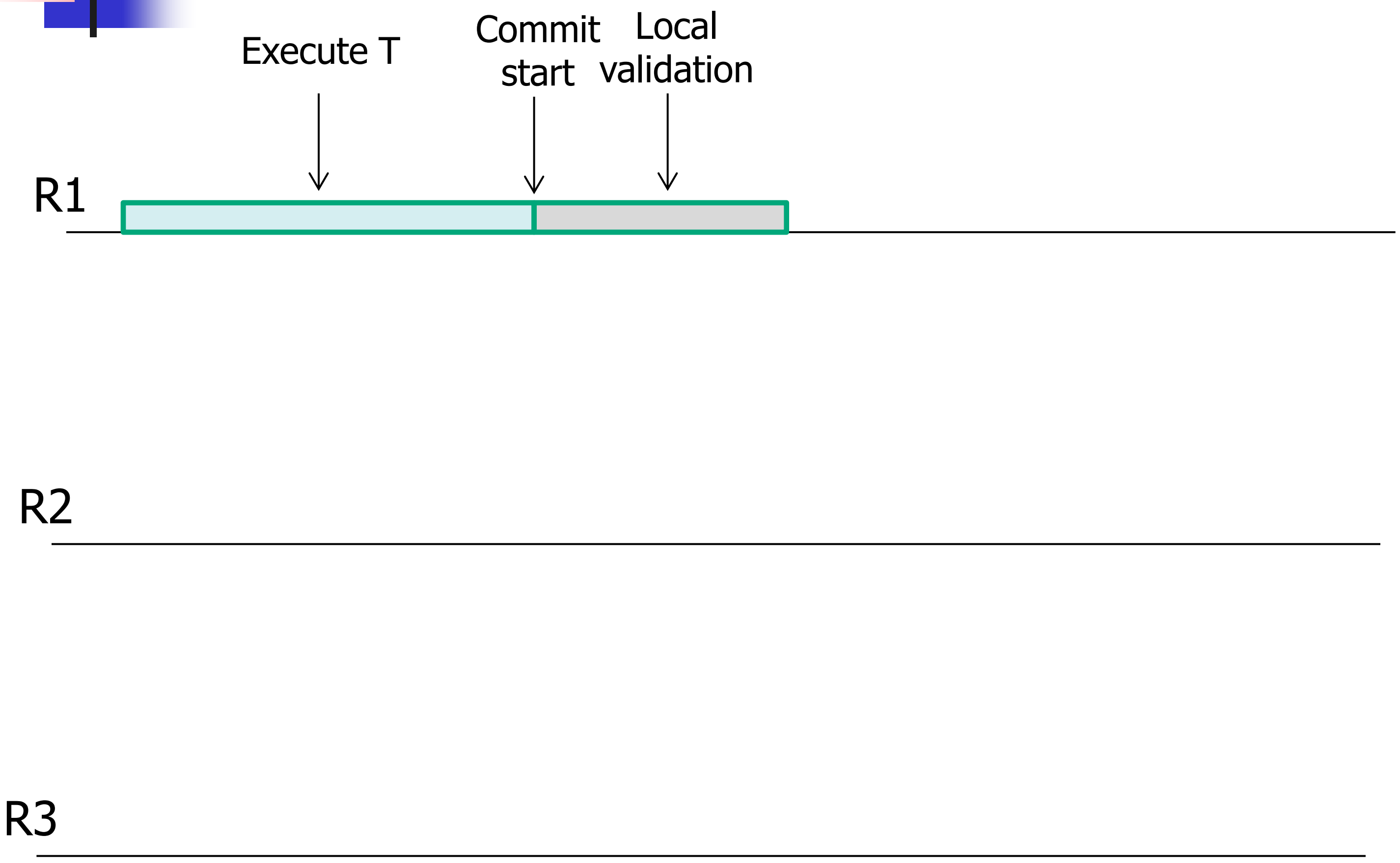
Coordination protocol



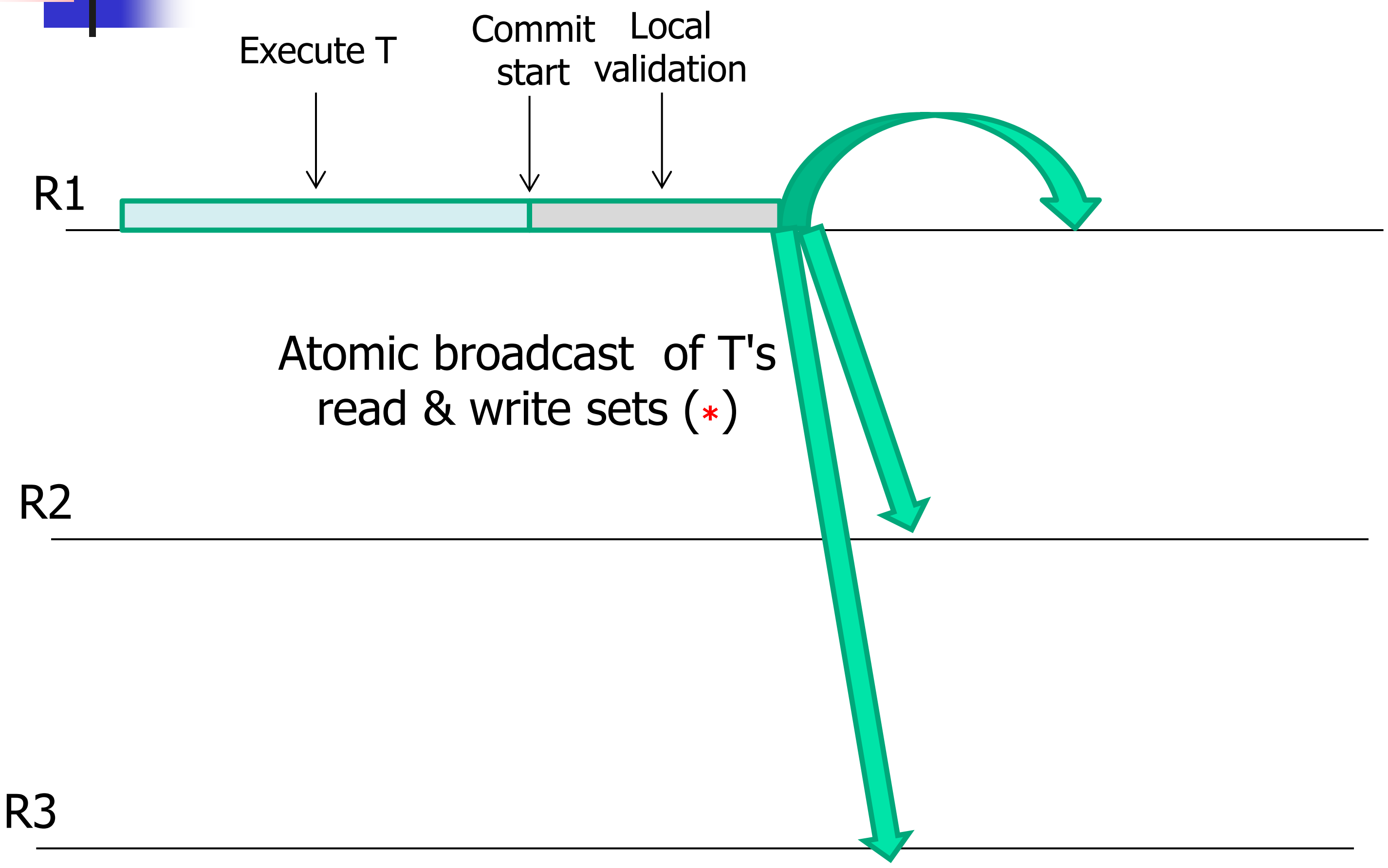
Coordination protocol



Coordination protocol

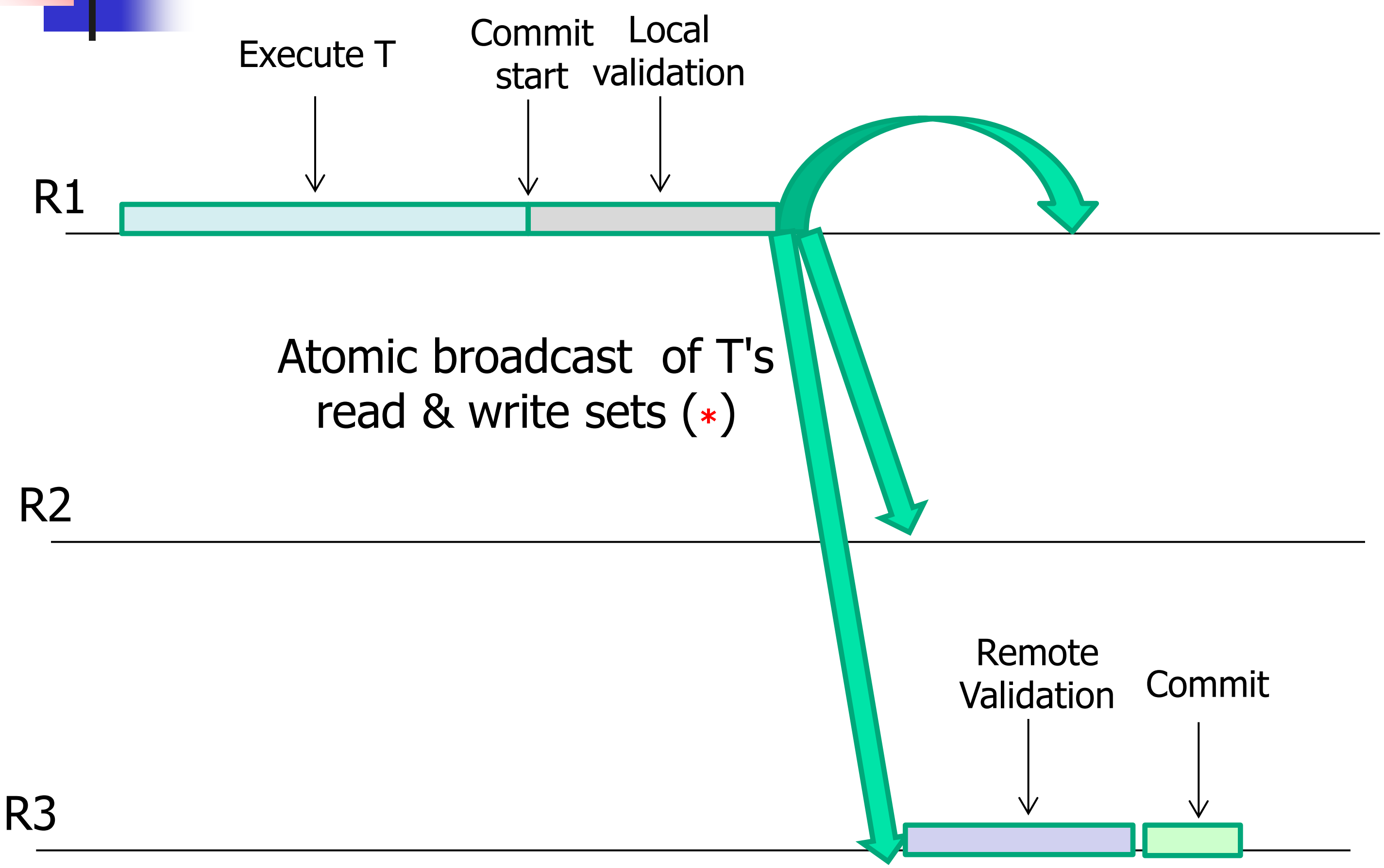


Coordination protocol

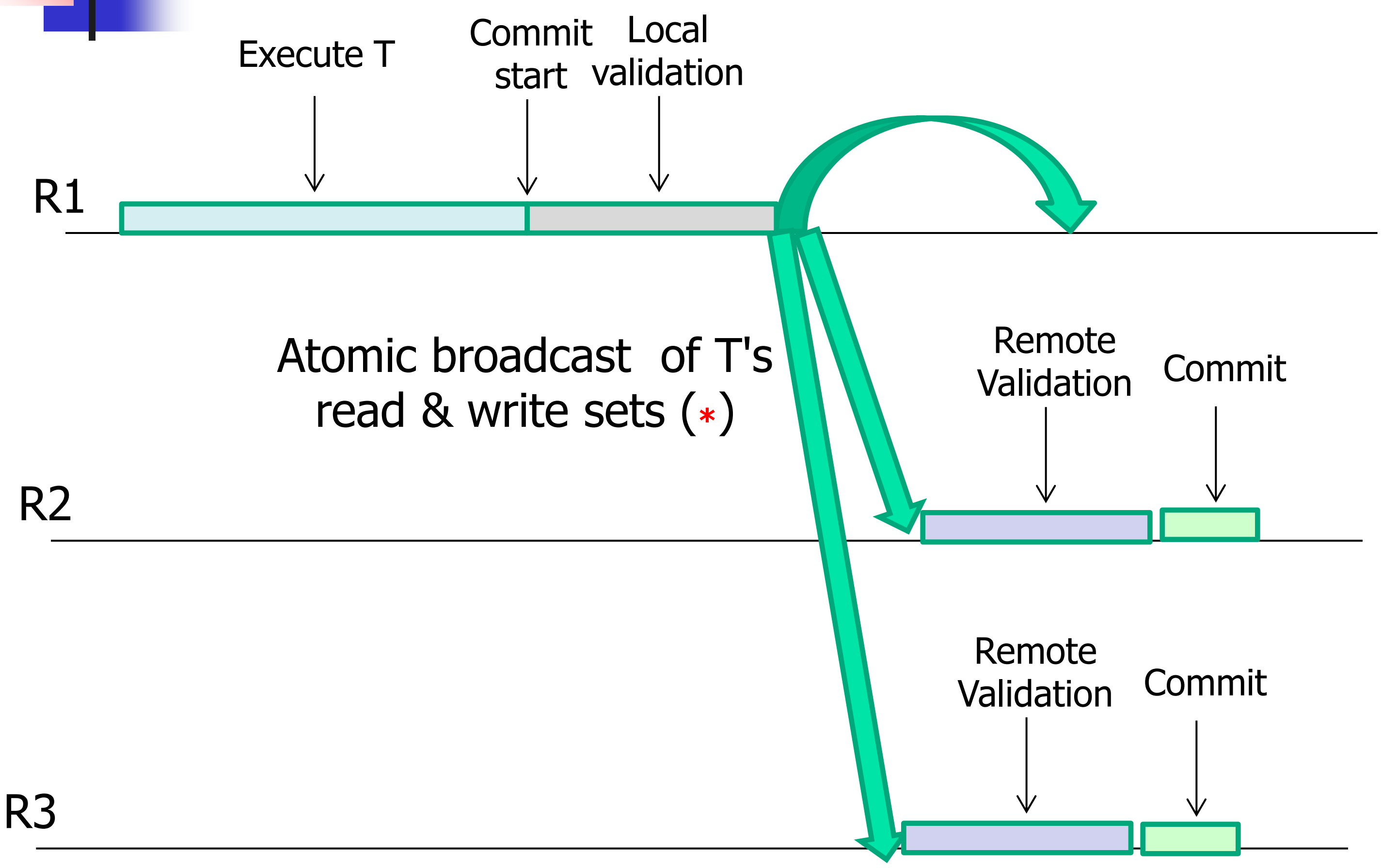


(*) In practice, only a Bloom filter of the read set is broadcast.

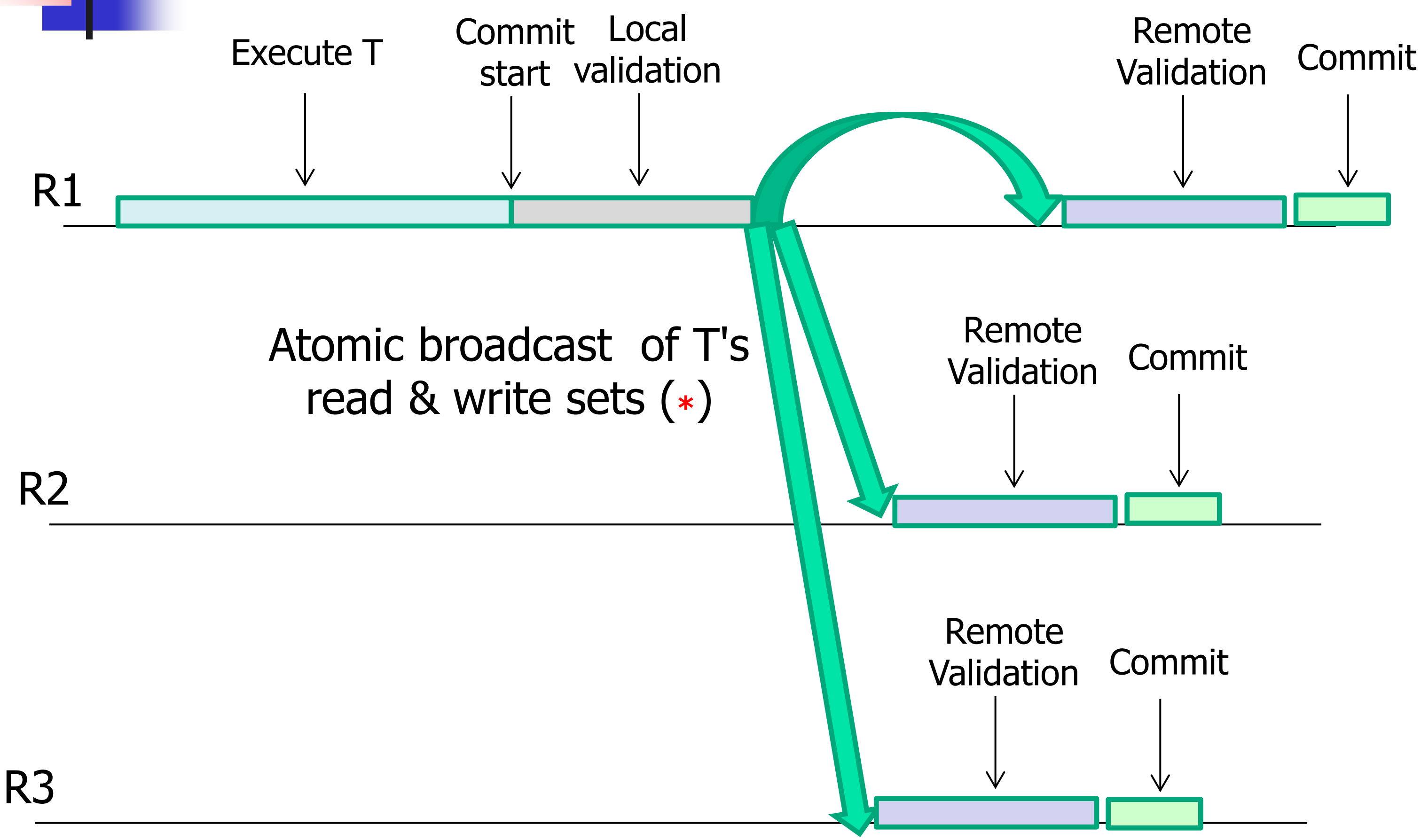
Coordination protocol



Coordination protocol



Coordination protocol



Asynchronous Lease-based Replication (ALC)

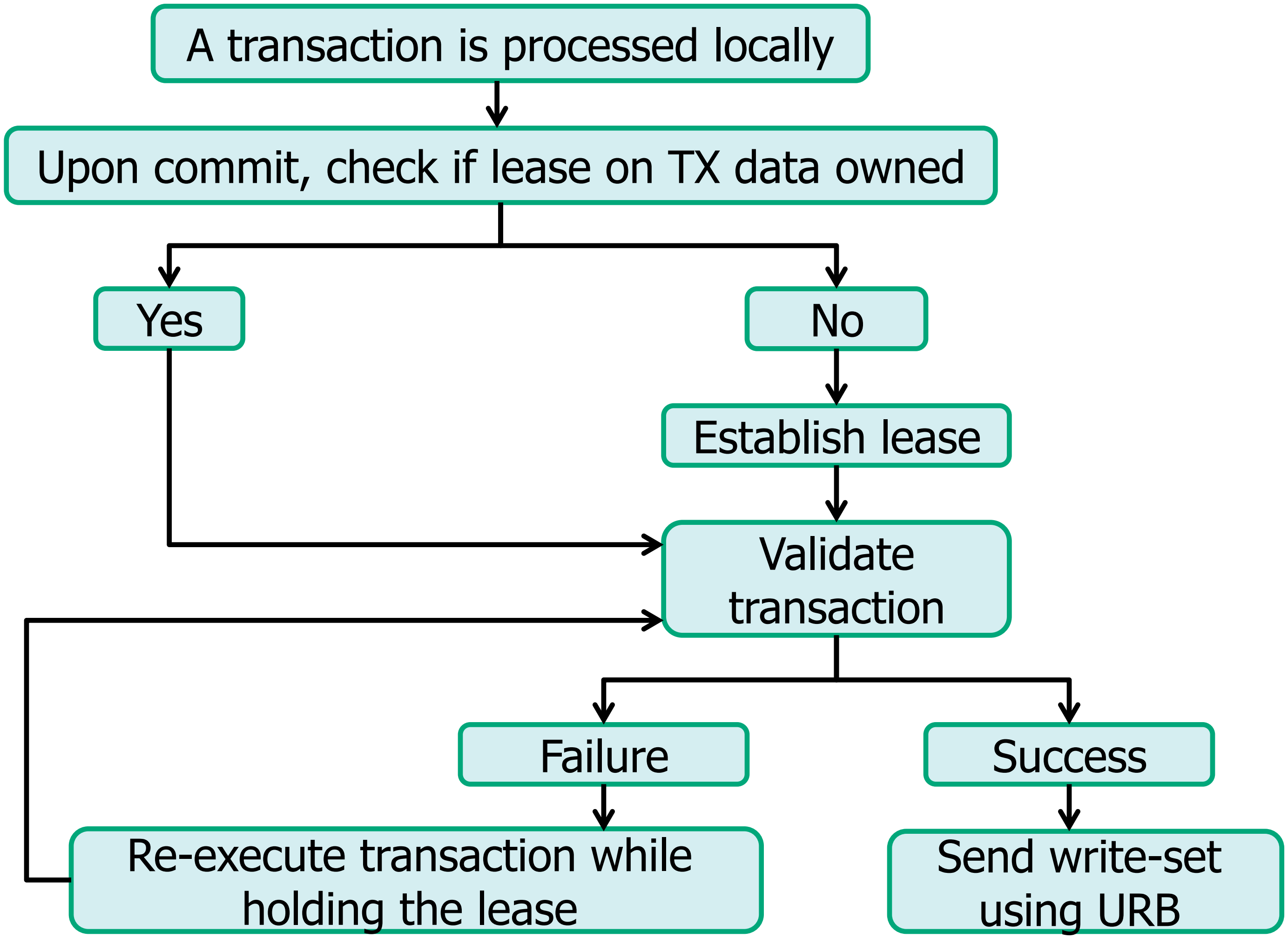
Carvalho, Romano & Rodrigues, Middleware`10

- Fully replicated DTM

- Nodes dynamically establish data ownership using leases
  - Write permission granted to lease-holder
  - Transactions sheltered from remote collisions

- Only write-set broadcast upon commit

ALC TX execution lifecycle



Preview for LiLAC-TM

- Fully replicated DTM, builds on ALC
  - First to support execution migration

- Performance gains for workloads exhibiting data locality

“Exploiting locality in lease-based replicated TM via task migration”, Hendler, Naiman, Peluso, Quaglia, Romano & Suissa

“Asynchronous lease-based replication of STM”, Carvalho, Romano & Rodrigues, Middleware‘10

Talk outline

- Preliminaries
- Multiprocessor contention management
  - Conflict resolution policies
  - TM schedulers
- Distributed TM (DTM)
  - Design space and principles
  - Example replicated DTM implementations
  - Contention management considerations

Contention management considerations

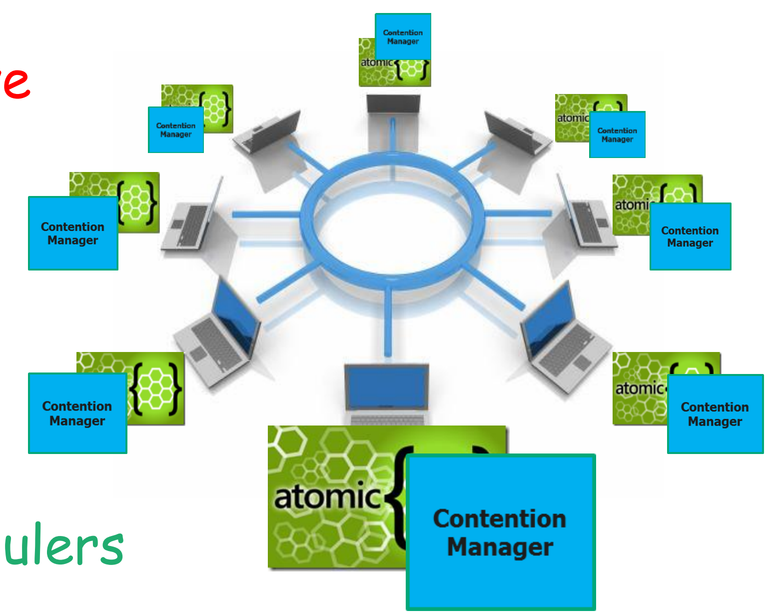
- Remote contention management coordination only at commit time
  - Some remote conflicts may be detected during execution
- Contention management for local threads managed by STM

Contention management considerations

- Remote contention management coordination only at commit time
  - Some remote conflicts may be detected during execution

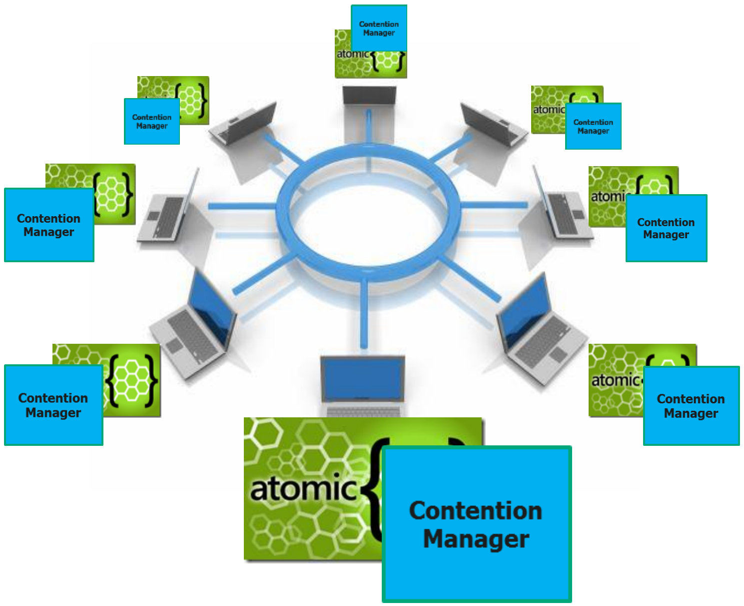
- Contention management for local threads managed by STM

Aggressive
Polite
Karma
greedy
Passive
Polka
TM schedulers



How to improve DTM contention mgmt?

- Better local & remote contention management synergy
  - Take lease ownership into consideration when resolving local conflicts?
- Serialize transactions across nodes?
- Migrate leases or TX execution?

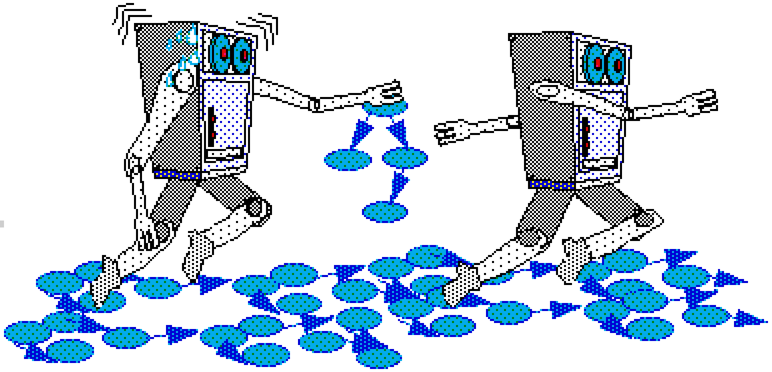


Summary

- Multiprocessor-TM contention management
  - "Conventional" conflict resolution policies
  - Scheduling-based (e.g. serialization) CM for high-contention workloads

- DTM uses local contention managers "as is"

- DTM performance may benefit from better local-remote contention management synergy



Thank you.