Object Based Transactional Memory

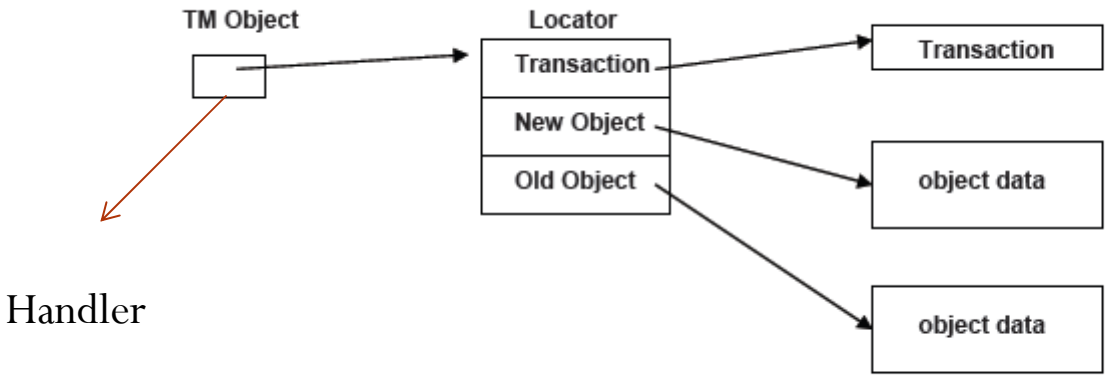
Ali Saoud

Introduction

- Resent trends go towards object based SMT because it's dynamic

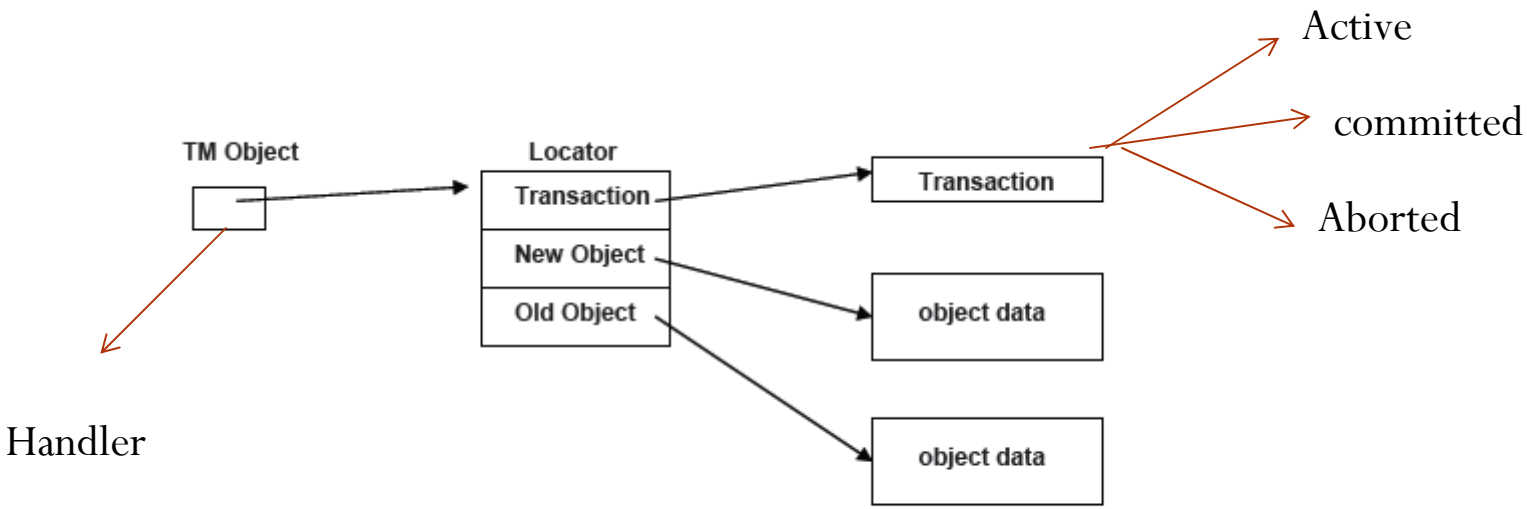
- Word-based STM systems are more suitable for data structures that may require concurrent access at a high level of granularity (e.g. multi-dimensional arrays).

Dynamic Software Transactional Memory (DSTM)



The locator is the main key to the design of the DSTM.
Every pointer goes through a level of indirection.

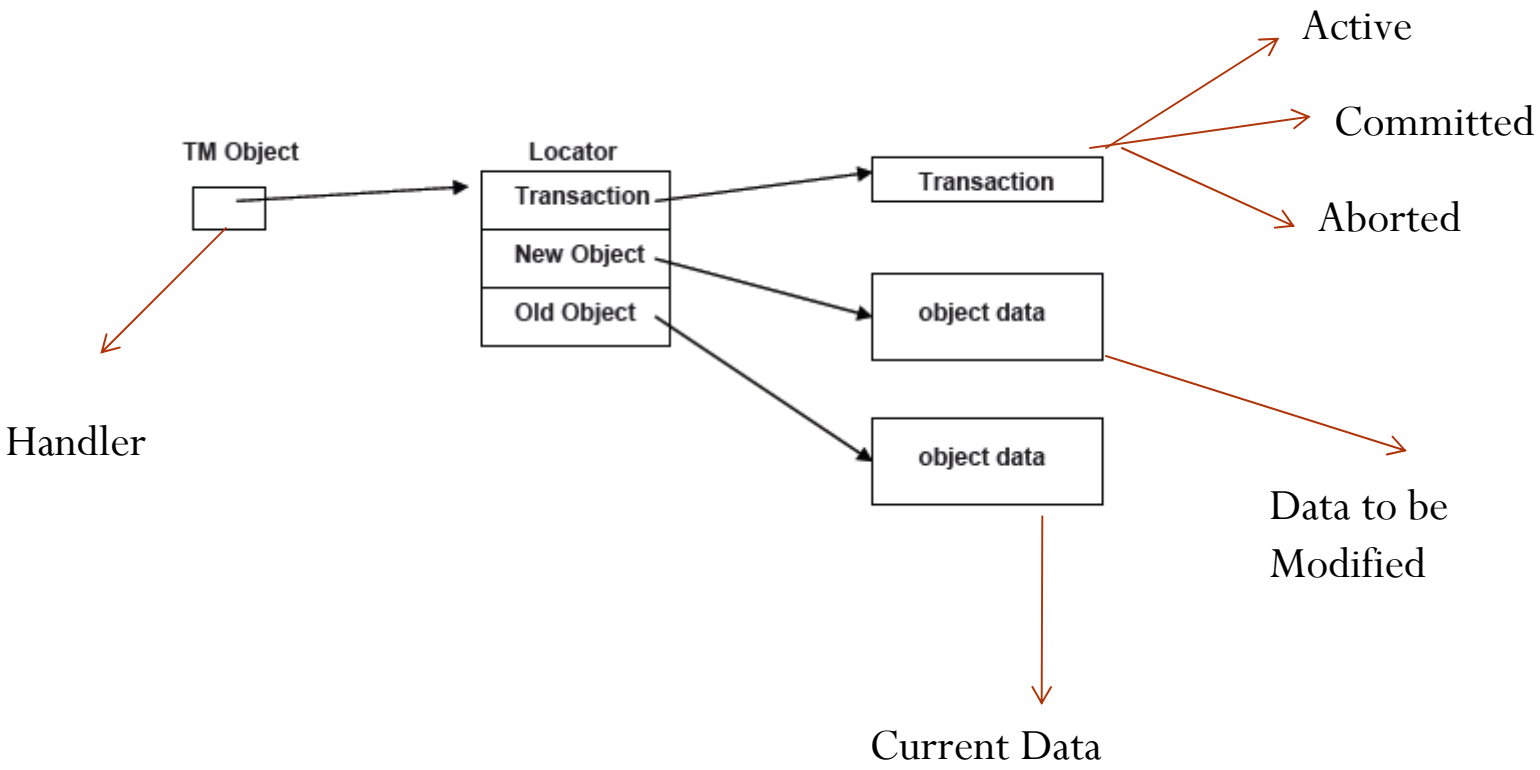
Dynamic Software Transactional Memory (DSTM)



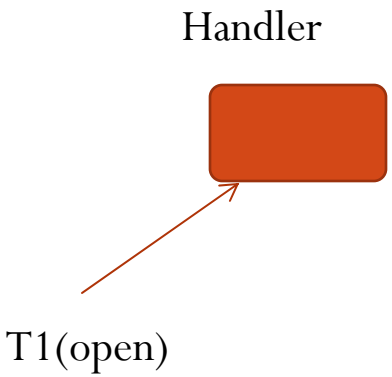
If the transaction is ACTIVE or ABORTED, the most recent valid version of the data object is the old version referenced by the locator.

If the transaction is COMMITTED, the most recent valid version of the data object is the new version referenced by the locator.

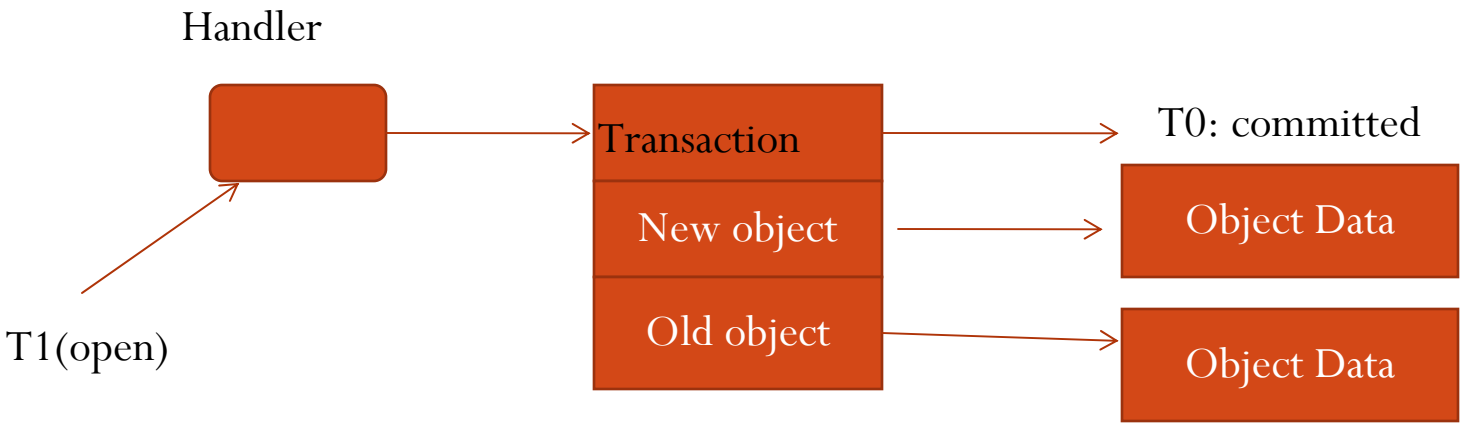
Dynamic Software Transactional Memory (DSTM)



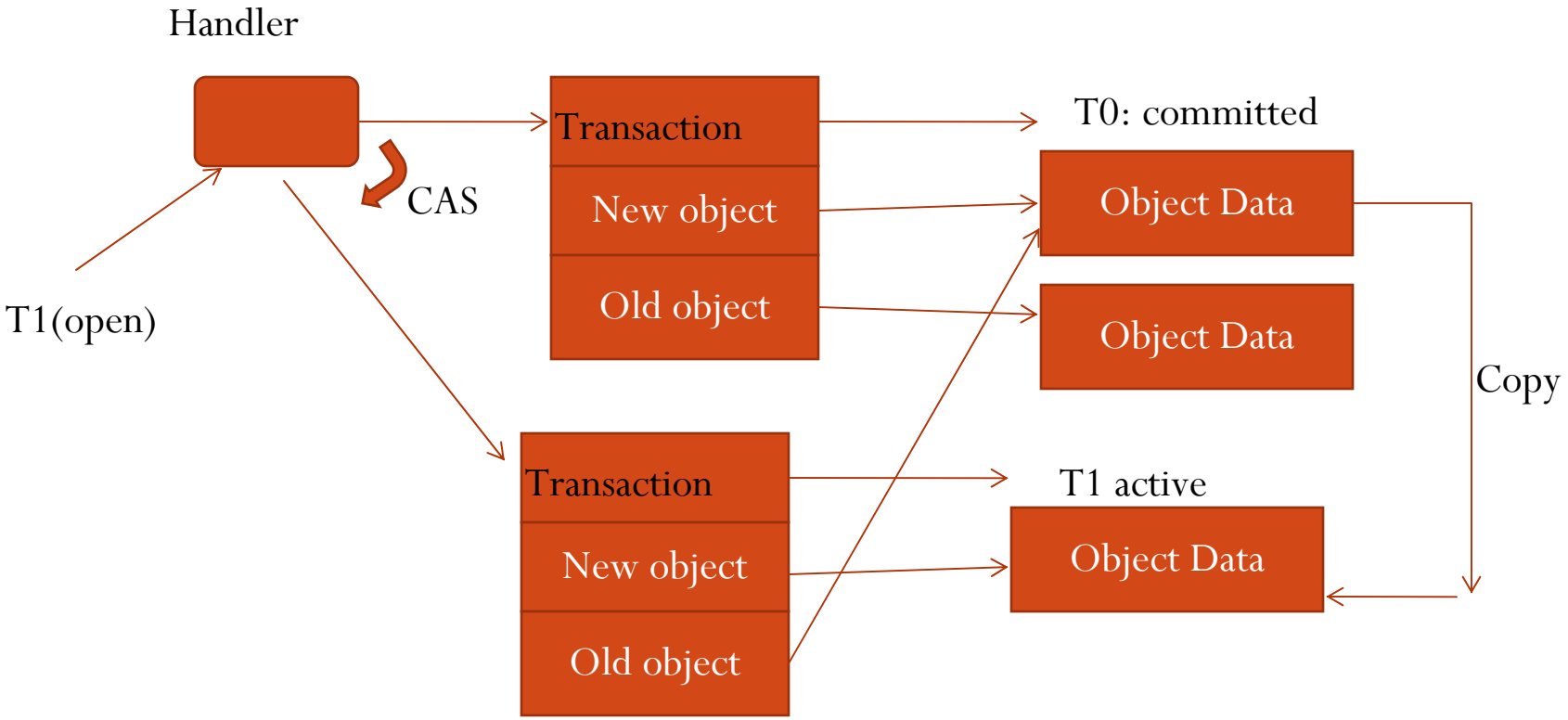
Open For Writing



Open For Writing

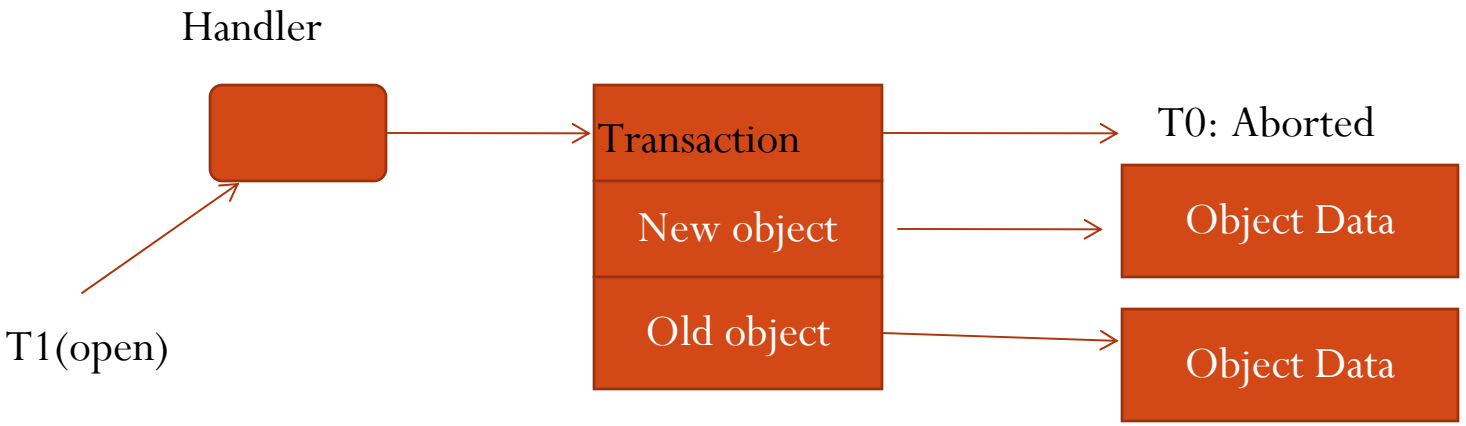


Open For Writing

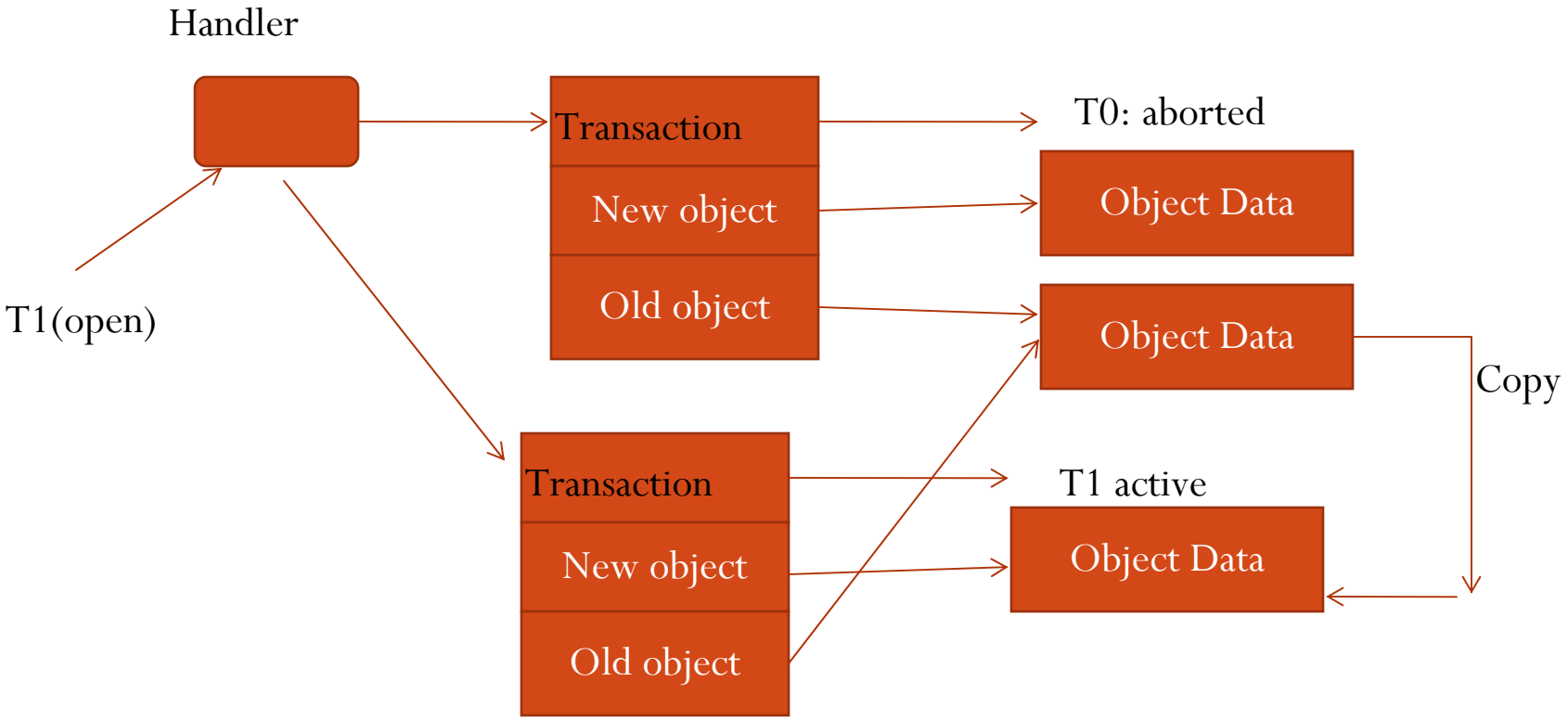


A successful CAS guarantees that the current transaction is visible to the entire concurrent system. A failure in CAS implies that some other transaction has opened (acquired) the TM Object in between.

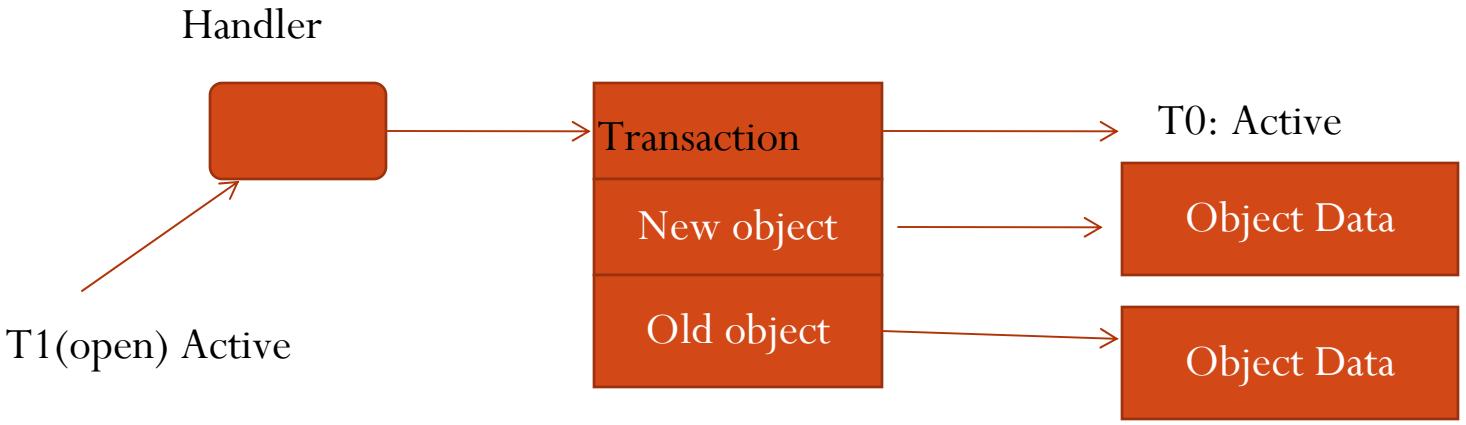
Open For Writing



Open For Writing



Open For Writing

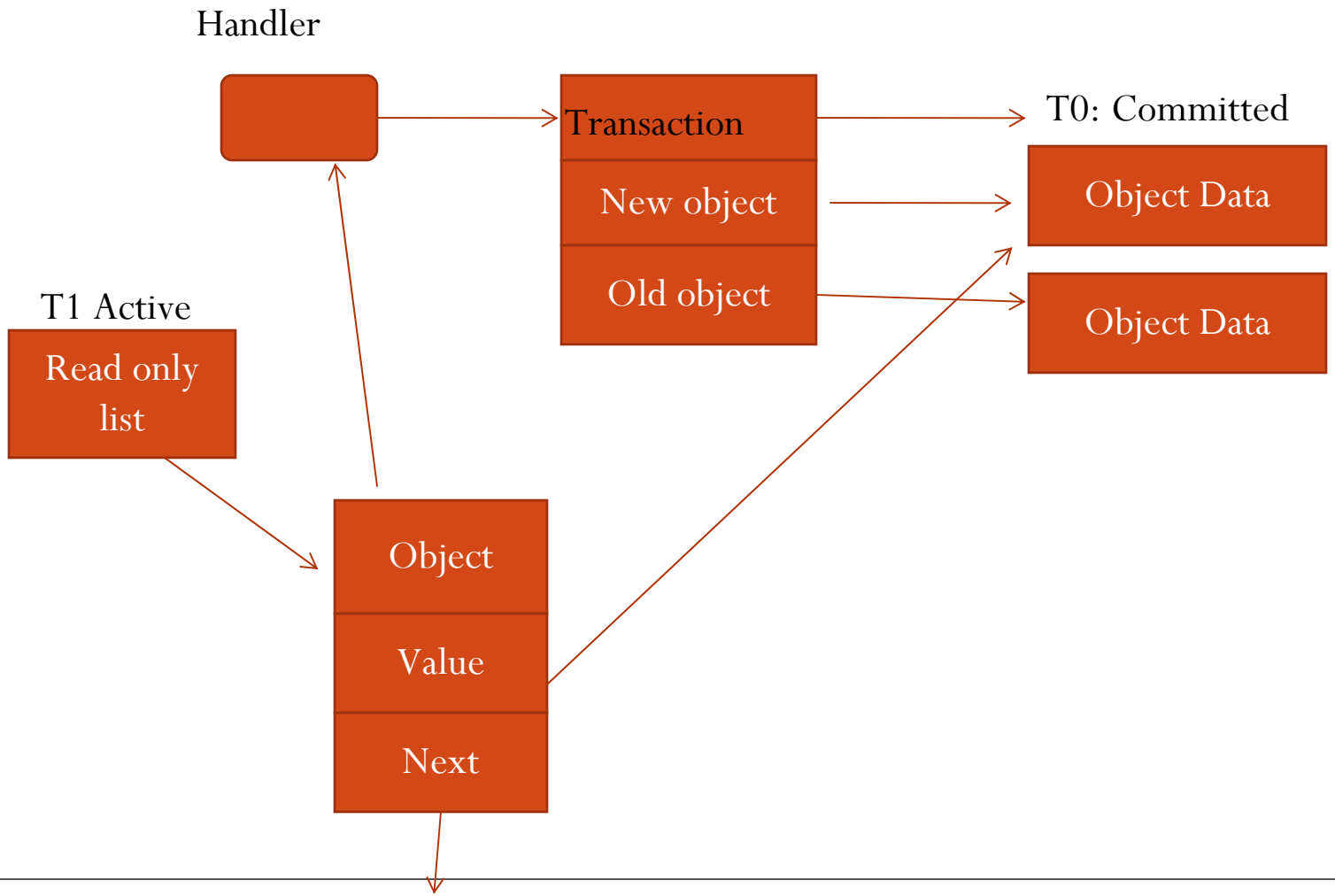


A transaction goes through a novel contention management protocol to decide whether to abort itself or the TM Object's current ACTIVE owner transaction.

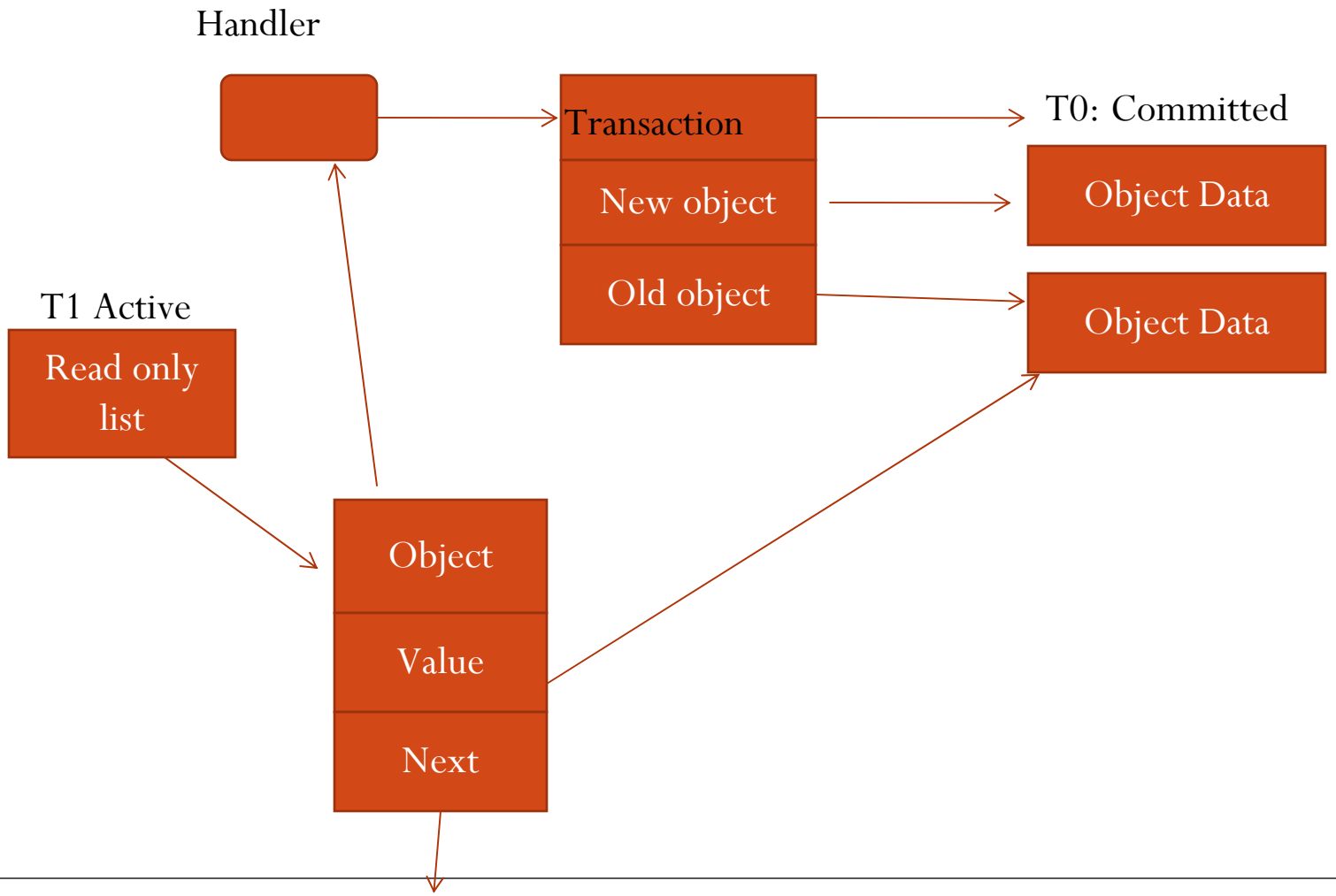
- aggressive –always/immediately aborts conflicting transaction
- polite –adaptive back-off

contention reduced by "early release": reference to object dropped before transaction commits and subsequent changes to the released object does not jeopardize consistency

Open For Reading



Open For Reading



FSTM

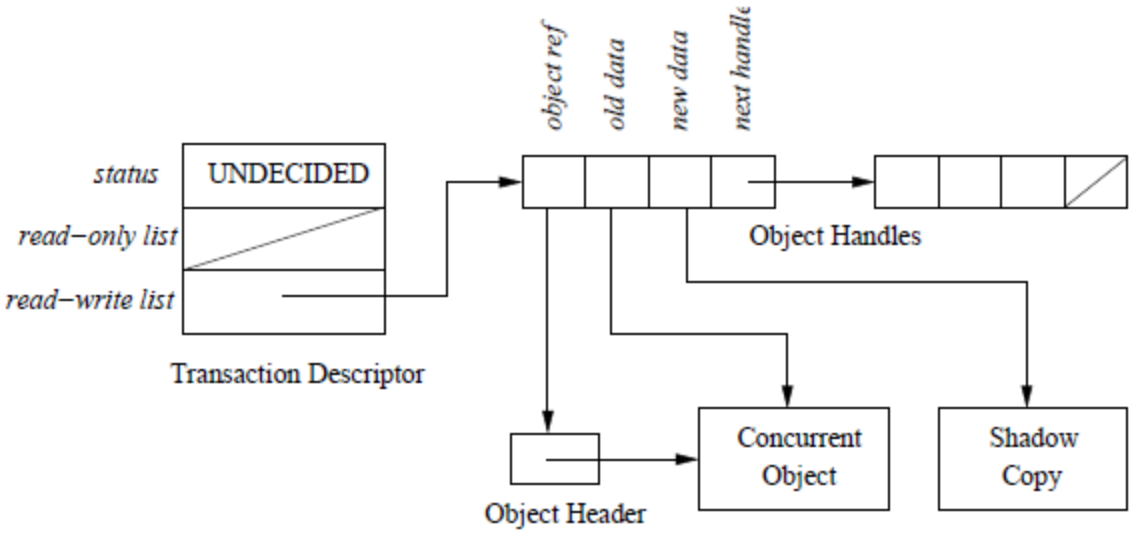


Figure 5: The basic Transactional Memory Structure in FSTM

Conflicts among transactions are detected and resolved at commit-time

FSTM

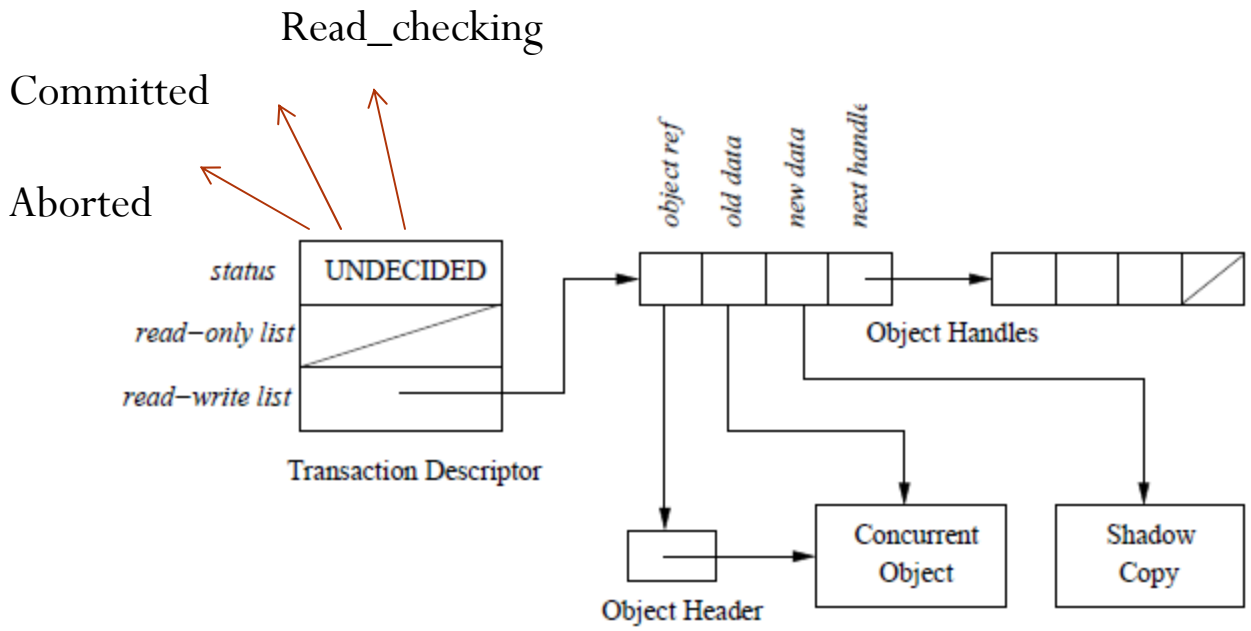


Figure 5: The basic Transactional Memory Structure in FSTM

Commit

- *Acquire*
  - Acquire each object in the read-write list in global total order using atomic CAS for each object
    - Abort if conflict with committed transaction detected
    - Help if conflict with uncommitted transaction detected

Commit 2

- *Read-checking*
  - Verify consistency of each object in the read-only list
    - Abort if change is detected in object held by Undecided transaction
    - If conflict detected with Read-checking transaction:
      - Help if other transaction precedes current transaction
      - Abort if current transaction precedes other transaction
- Release acquired transactions

Comparison

Characteristic	System			
	STM-1	WSTM	DSTM	FSTM
Strong/Weak Isolation	N/A	Weak	Weak	Weak
Granularity	Word	Word	Object	Object
Direct/Deferred Update	Direct	Deferred (update in place)	Deferred (clone replacement)	Deferred (clone replacement)
Concurrency Control	Pessimistic	Optimistic	Optimistic	Optimistic
Synchronization	Lock-free	Obstruction-free	Obstruction-free	Lock-free
Conflict Detection	Early	Late	Early	Late
Inconsistent Reads	None	Toleration	Validation	Validation
Conflict Resolution	Helping	Helping/aborting	Contention manager	Abort
Nested Transactions		Flattened	Flattened	Closed
Exceptions		Terminate		