COMP207 Database Development

Tutorial 4 (Week 6)

Timestamping

Timestamp Protocol

- Instead of using a locking protocol, the timestamp protocol assigns a variable (‘the timestamp’) to each transaction to control which transactions have access to the data items
- The timestamp may be the value of the system clock at the time the transaction enters the system, or it may be something simpler
- A simpler algorithm assigns a number to the transaction. The only criteria is that the next transaction to enter the system is given a bigger number than the one before it

Timestamp Protocol

- Transaction1 enters the system

Timestamp Protocol

- Transaction1 enters the system

T1

Timestamp Protocol

... And is given a ‘timestamp’

(here the variable is a number rather than using the system clock)

T1

Timestamp Protocol

... And is given a ‘timestamp’

(here the variable is a number [1] rather than using the system clock)

Timestamp Protocol

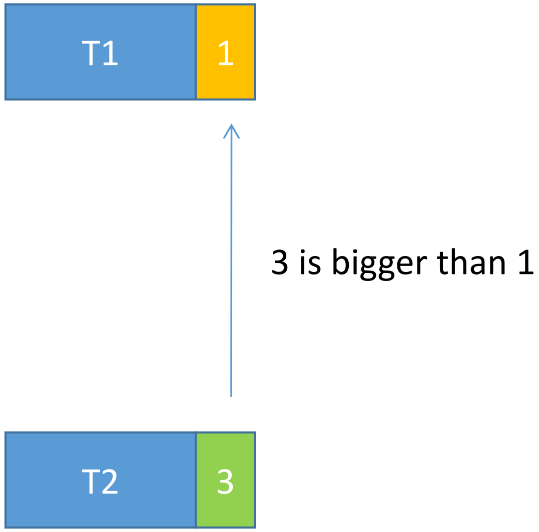
- Transaction2 enters the system and is given a ‘timestamp’ (here the variable is a number [3] rather than using the system clock)

Timestamp Protocol

- The important thing is that the value of the timestamp for T2 is bigger than that for T1

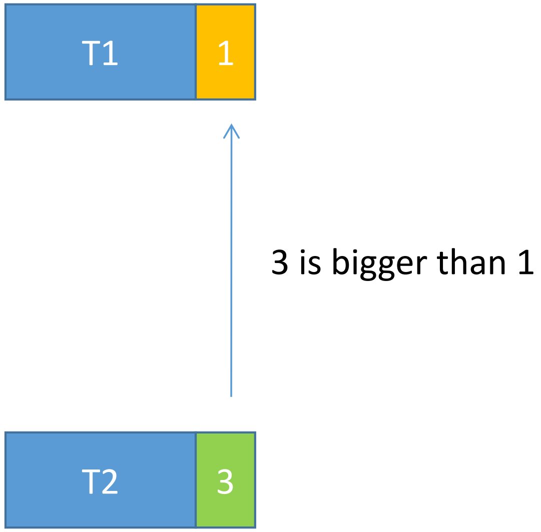
Timestamp Protocol

- The important thing is that the value of the timestamp for T2 is bigger than that for T1



Timestamp Protocol

- If a third transaction (T3) were to enter the system, it would be given a timestamp and that timestamp MUST be bigger than the one given to T2 (because T2 entered the system before T3)

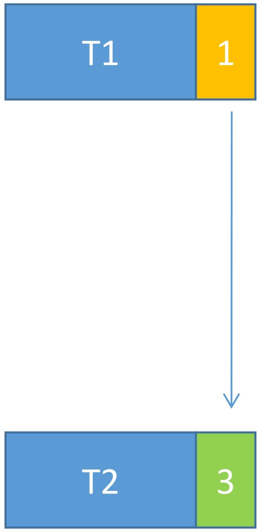


Timestamp Protocol

- By using these ascending numbers, we can identify transactions as being ‘younger’ or ‘older’ than each other based on the size of their timestamp value

Timestamp Protocol

- As T1 has a timestamp value less than T2, then T1 is the 'older' transaction – it has a lower timestamp value (and so has been in the system longer) than T2
- This means here that T2 is the 'younger' transaction



1 is less than 3, so T1 is the older transaction

Timestamp Protocol

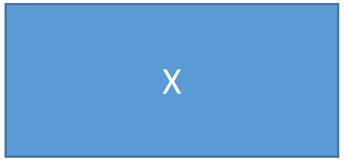
- Data items also have a timestamp value assigned to them
- In fact they have TWO timestamp values
  - One for a Read Timestamp (RT)
  - One for a Write Timestamp (WT)
  - They start with values of zero (0) for RT and WT and then get these values altered to the values of the transactions that access them
    - Our transactions have TS[T1]=1 and TS[T2]=3

Timestamp Protocol

- T1 and T2 are accessing two data items

Timestamp Protocol

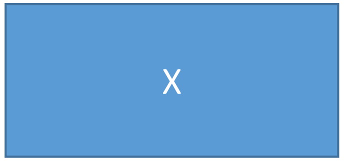
- T1 and T2 are accessing two data items - X



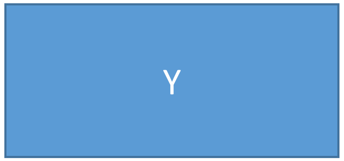
X

Timestamp Protocol

- T1 and T2 are accessing two data items - X

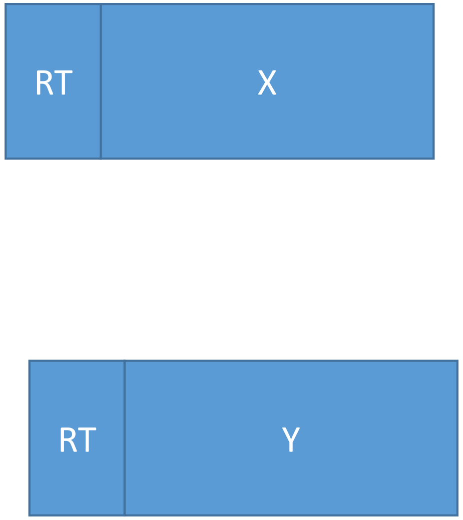


- ... and Y



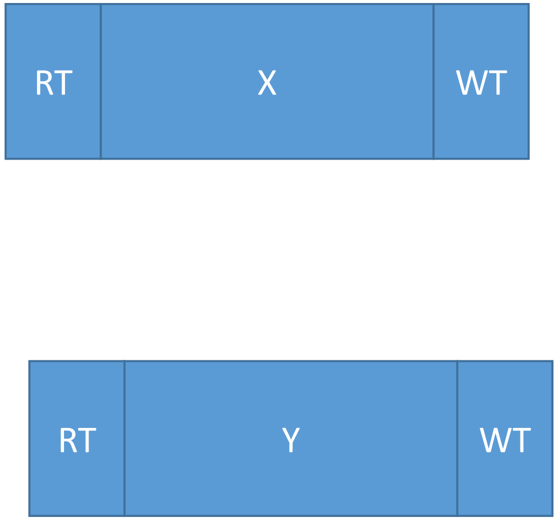
Timestamp Protocol

- And they are assigned timestamp values for read operations
    - RT(X), RT(Y)



Timestamp Protocol

- ... And for write operations
  - WT(X), WT(Y)

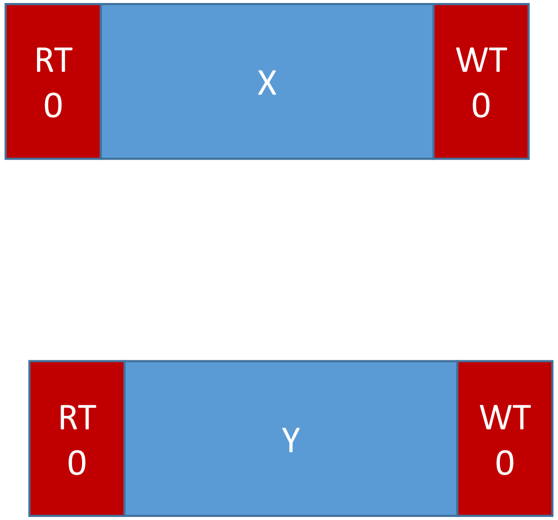


RT X WT

RT Y WT

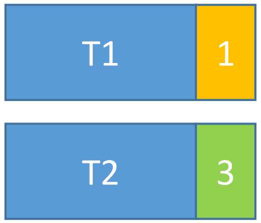
Timestamp Protocol

- And at the start of the schedule, they are all initialised to zero (0)



Timestamp Protocol

- We now have two transactions with an assigned timestamp such that:
- TS[T1] = 1
- TS[T2] = 3

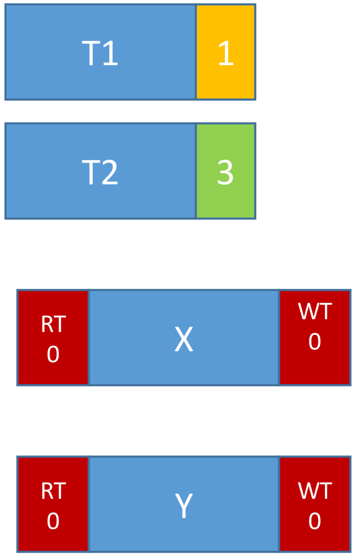


T1 1

T2 3

Timestamp Protocol

- ... And two data items, X and Y such that:
- RT(X) = 0
- WT(X) = 0
- RT (Y) = 0
- WT (Y) = 0



Timestamp Protocol

- ... Or written as a table

time	Op.	T1	T2	RT(X)	WT(X)	RT(Y)	WT(Y)
		1	3	0	0	0	0
1							
2							
3							
4							
5							
6							

Performing a read on an item

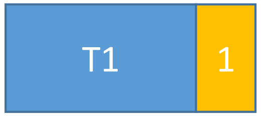
- T1 wants to read item (Y)
- T1 can only read Y if the timestamp value of T1 is greater than the Write Transaction value of the data item

- If TS[T1] >= WT(Y) then
  - Read
  - Set read_TS(Y) to larger of TS(T) and current read_TS(Y)

- If the comparison above fails (TS[T1] < WT(Y)) then a younger transaction has already written Y – abort T1 and reject the operation

Performing a read on an item

- T1 wants to read ...

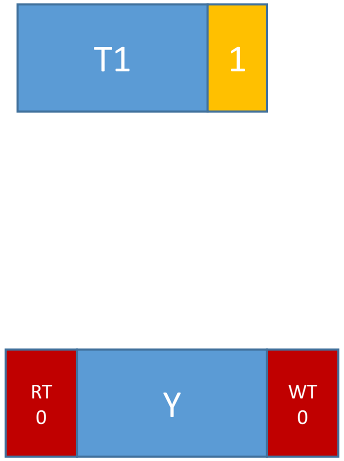


T1

1

Performing a read on an item

- T1 wants to read item (Y)

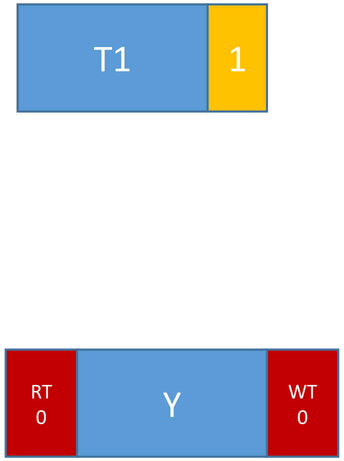


T1 1

RT 0 Y WT 0

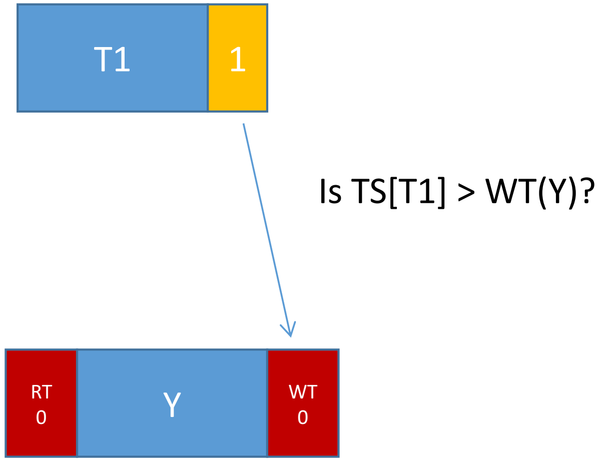
Performing a read on an item

- T1 can only read Y if its timestamp is >= than Write Timestamp of Y (If TS[T1] >= WT(Y) then read)



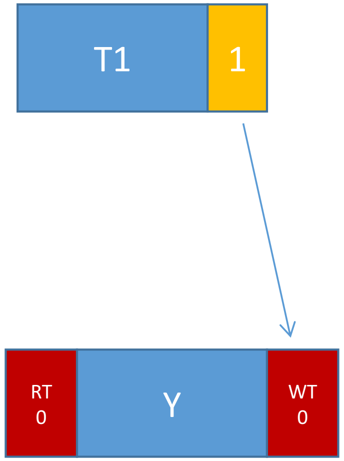
Performing a read on an item

- T1 can only read Y if its timestamp is >= than Write Timestamp of Y



Performing a read on an item

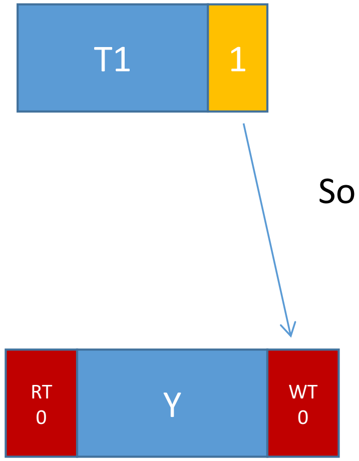
- T1 can only read Y if its timestamp is >= than Write Timestamp of Y



YES – the timestamp of the transaction is greater than the write timestamp of the data item

Performing a read on an item

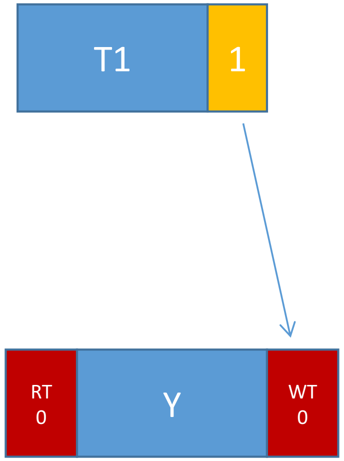
- T1 can only read Y if its timestamp is >= than Write Timestamp of Y



So T1 is allowed to read item Y

Performing a read on an item

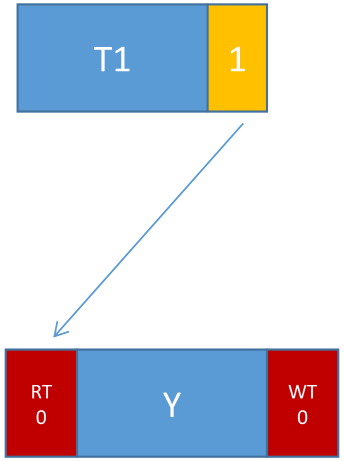
- T1 can only read Y if its timestamp is >= than Write Timestamp of Y



Item Y must now be flagged as having been read by T1

Performing a read on an item

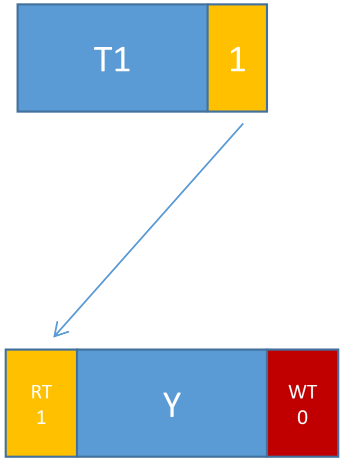
- T1 can only read Y if its timestamp is >= than Write Timestamp of Y



We do this by setting the Read Transaction value of Y to the timestamp value of T1 (because T1 has now read item Y)

Performing a read on an item

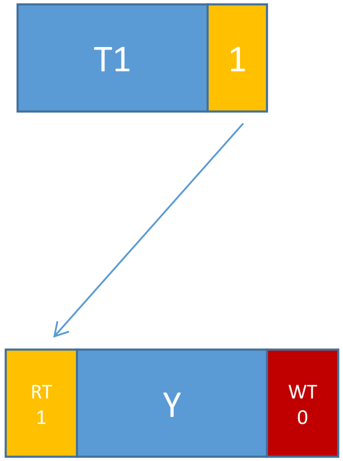
- T1 can only read Y if its timestamp is >= than Write Timestamp of Y



We do this by setting the Read Transaction value of Y to the timestamp value of T1 (because T1 has now read item Y)

Performing a read on an item

- T1 can only read Y if its timestamp is >= than Write Timestamp of Y



Note the timestamp value
of T1 does not change

Performing a write on an item

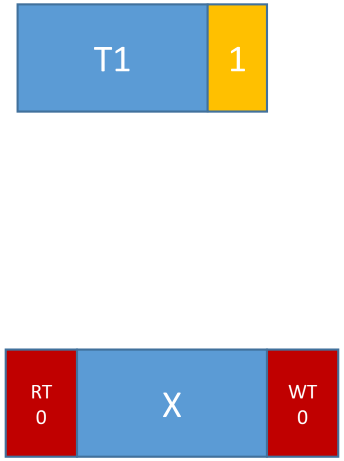
- T1 wants to write item (X)
- T1 can only write X if the timestamp value of T1 is greater than or equal to the Write Transaction value of the data item AND greater than or equal to the Read Transaction value of the data item
    - If (TS[T1] > =WT(X)) AND (TS[T1] >= RT(X)) then write
- If the above comparison does not hold, then a younger transaction has read X, so abort T1 and reject the operation

Performing a write on an item

- T1 wants to write ...

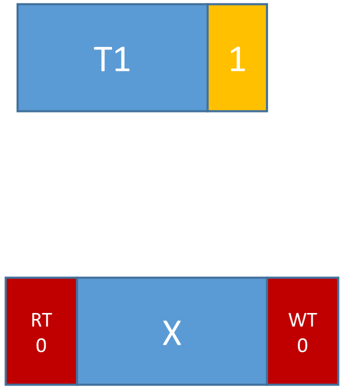
Performing a write on an item

- T1 wants to write item (X)



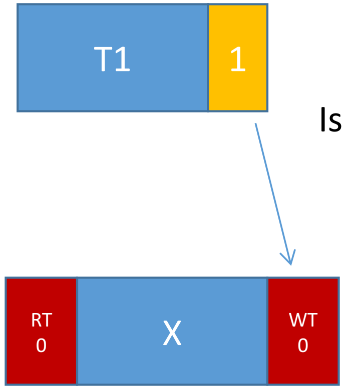
Performing a write on an item

- T1 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X



Performing a write on an item

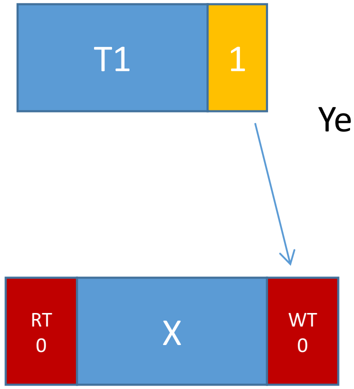
- T1 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X



Is TS(T1) >= WT(X)?

Performing a write on an item

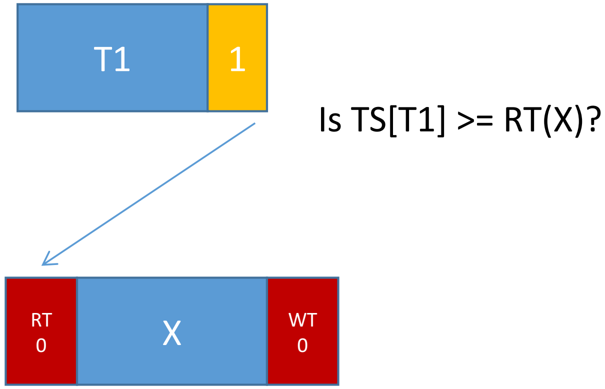
- T1 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X



Yes – TS[T1] >= WT(X)

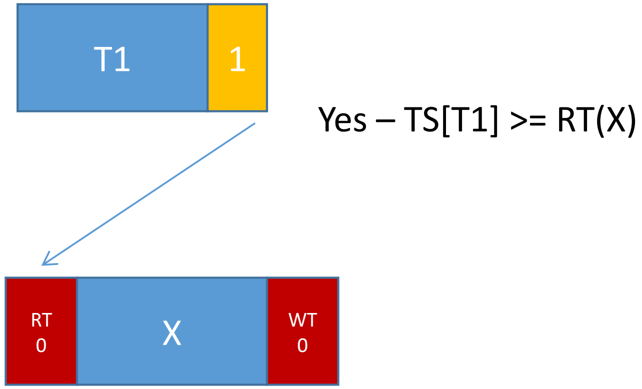
Performing a write on an item

- T1 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X



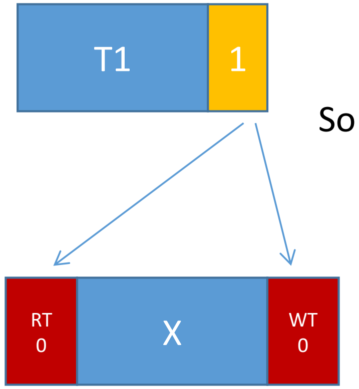
Performing a write on an item

- T1 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X



Performing a write on an item

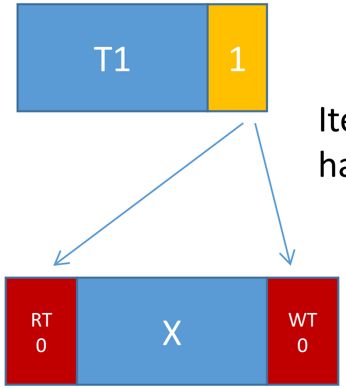
- T1 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X



So T1 is allowed to write item X

Performing a write on an item

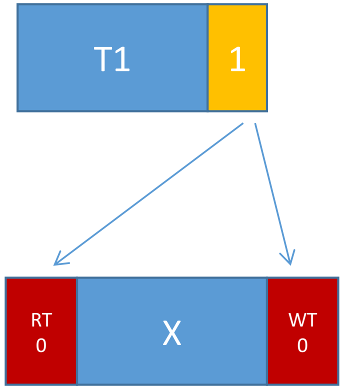
- T1 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X



Item X must now be flagged as having been written by T1

Performing a write on an item

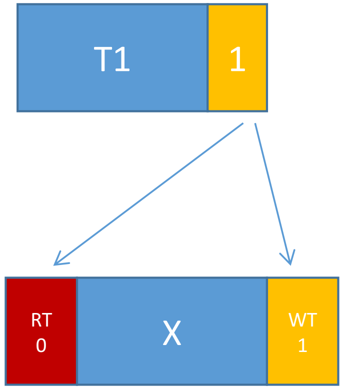
- T1 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X



We do this by setting the Write Transaction value of X to the timestamp value of T1 (because T1 has now written item X)

Performing a write on an item

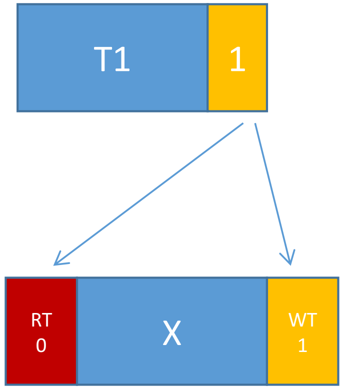
- T1 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X



We do this by setting the Write Transaction value of X to the timestamp value of T1 (because T1 has now written item X)

Performing a write on an item

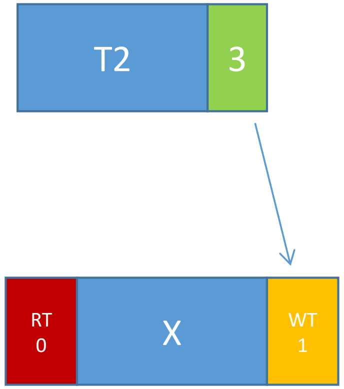
- T1 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X



Note the timestamp value of T1 does not change

Performing a write on an item

- T2 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X:

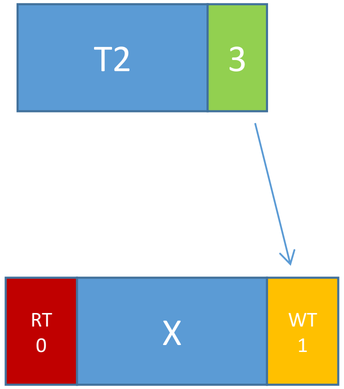


Item X has been previously written by T1 and now T2 requests a write operation on it:

Is TS[T2] >= WT(X)?

Performing a write on an item

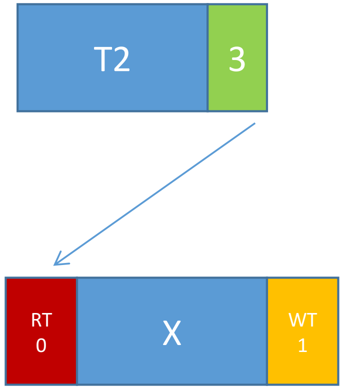
- T2 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X:



YES: TS[T2] >= WT(X)

Performing a write on an item

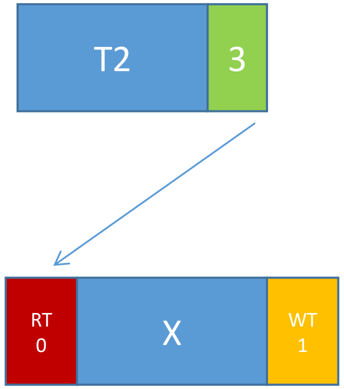
- T2 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X:



Is TS[T2] >= RT(X)?

Performing a write on an item

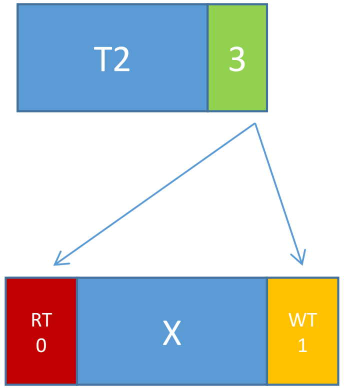
- T2 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X:



YES – TS[T2] >= RT(X)

Performing a write on an item

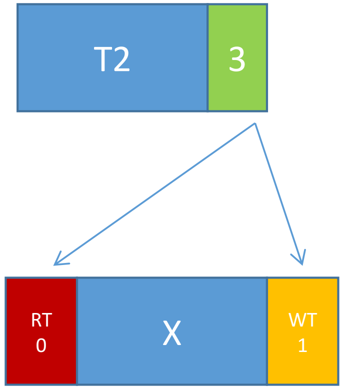
- T2 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X:



So T2 can write item X

Performing a write on an item

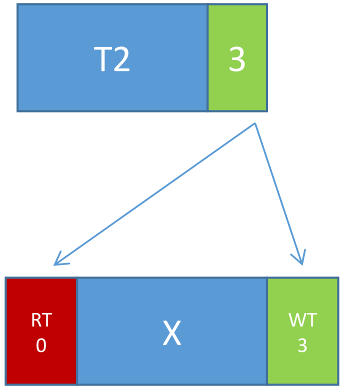
- T2 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X:



Item X must now be flagged as having been written by T2 (thus overwriting the change written previously by T1)

Performing a write on an item

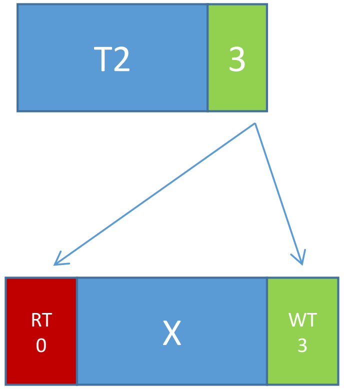
- T2 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X:



Item X must now be flagged as having been written by T2 (thus overwriting the change written previously by T1)

Performing a write on an item

- T2 can only write X if its timestamp is >= Write Timestamp of X AND >= Read Timestamp of X:

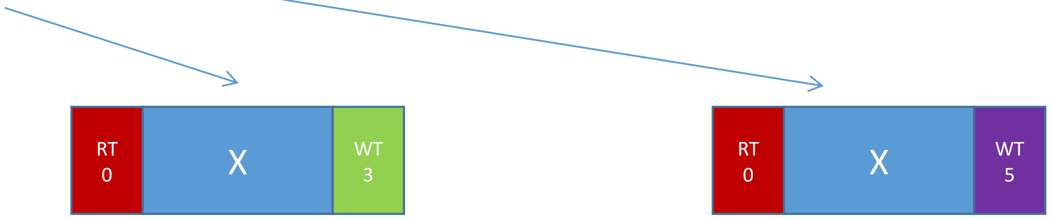


Note the timestamp value of
T2 does not change

What If …

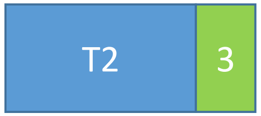
- … A third transaction is in the system

- This conforms with the algorithm as T3 was the last transaction to enter the system, and is given a timestamp of [5] which is greater than T1 and T2 and therefore T3 is the youngest transaction
- It too has been accessing data item X with a write request such that WT(X) goes from this to this



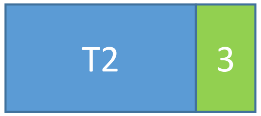
What If ...

Suppose T2 ...



What If …

Suppose T2 …

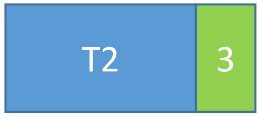


T2 3

Wants to write item X

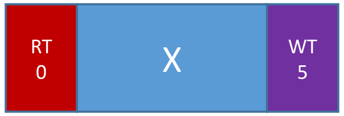
What If ...

Suppose T2 ...



T2 3

Wants to write item X (which has been written by T3)



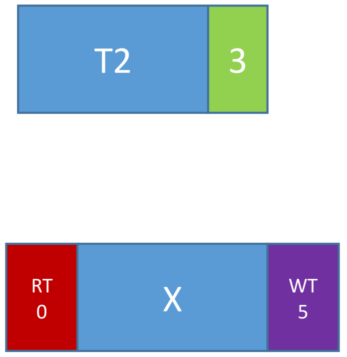
RT 0

X

WT 5

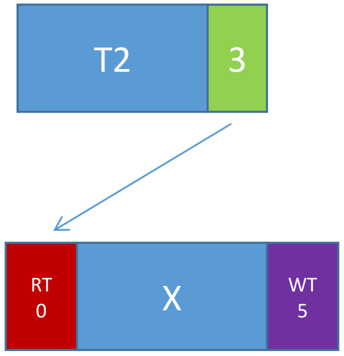
What If …

- T2 may only write item X if the timestamp value of T2 is greater than or equal to the Read Timestamp of item X and is greater than or equal to the Write Timestamp of item X



What If …

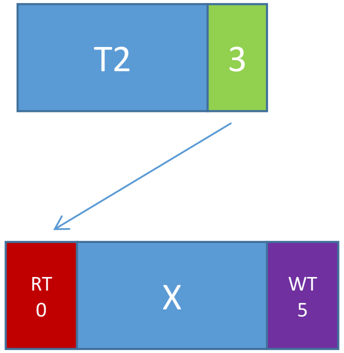
- T2 may only write item X if the timestamp value of T2 is greater than or equal to the Read Timestamp of item X and is greater than or equal to the Write Timestamp of item X



Is TS(T2) >= RT(X)?

What If …

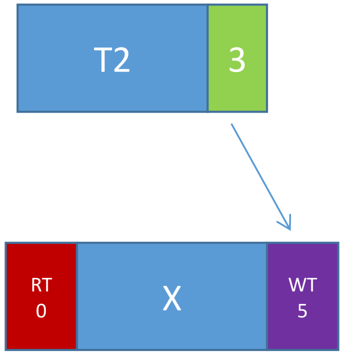
- T2 may only write item X if the timestamp value of T2 is greater than or equal to the Read Timestamp of item X and is greater than or equal to the Write Timestamp of item X



YES , TS(T2) >= RT(X)
… So far, so good

What If …

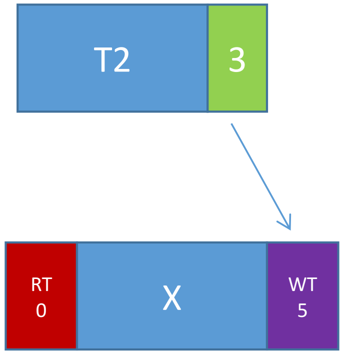
- T2 may only write item X if the timestamp value of T2 is greater than or equal to the Read Timestamp of item X and is greater than or equal to the Write Timestamp of item X



Is TS(T2) >= WT(X)?

What If …

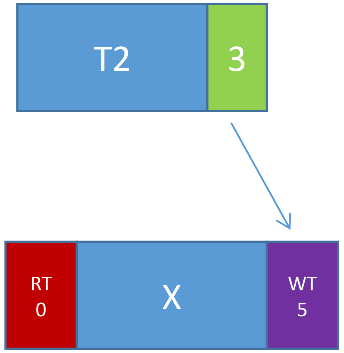
- T2 may only write item X if the timestamp value of T2 is greater than or equal to the Read Timestamp of item X and is greater than or equal to the Write Timestamp of item X



No - TS(T2) < WT(X)

What If …

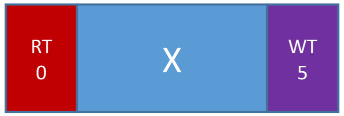
- T2 may only write item X if the timestamp value of T2 is greater than or equal to the Read Timestamp of item X and is greater than or equal to the Write Timestamp of item X



T2 will have to be restarted and allowed to re-enter the system with a new timestamp (which must be greater then the largest transaction timestamp that is currently being executed)

What If …

- T2 may only write item X if the timestamp value of T2 is greater than or equal to the Read Timestamp of item X and is greater than or equal to the Write Timestamp of item X
- Item X retains its current state (until another transaction changes it)



Timestamp Ordering

- End of Demonstration