CS4224/CS5424 Lecture 6
Distributed Concurrency Control

Single/Multi-Partition Transactions

- Single-partition transaction: a distributed transaction that access/update items from exactly one site
- Multi-partition transaction: a distributed transaction that access/update items from more than one site
- **Example**: Site A = $\{x\}$, Site B = $\{y\}$
  - $T_1 = R_1(x), W_1(x), R_1(y), W_1(y)$
  - $T_2 = R_2(x), R_2(y)$
  - $T_3 = R_3(x), W_3(x)$

Distributed Transactions

- Transaction originating site - site where Xact is initiated
- Transaction coordinator (TC) - transaction manager (TM) at originating site
- TC of a distributed Xact T coordinates with other TMs to execute T at multiple sites

Example

- DDBMS: Site A = $\{x\}$, Site B = $\{y\}$, Site C = $\{z\}$
- Distributed Xact T_1 is initiated at Site A:

$$T_1 : \quad R_1(x), R_1(y), W_1(x), R_1(z), W_1(z)$$

- T_1 is executed as 3 local transactions:
  - $T_{1,A}$: $R_1(x), W_1(x)$
  - $T_{1,B}$: $R_1(y)$
  - $T_{1,C}$: $R_1(z), W_1(z)$
- Originating site: Site A
- Transaction coordinator: TM at Site A

Local Schedule

- **Local schedule** = transaction schedule at a local site
- **Example**:

$T = \{T_1, T_2\}$
Site A: x
Site B: y

T_1: Read(x)
x = x - 100
Write(x)
Read(y)
y = y + 100
Write(y)

T_2: Read(x)
Read(y)

- Possible local schedules:
  - S_A: $R_1(x), W_1(x), R_2(x)$
  - S_B: $R_2(y), R_1(y), W_1(y)$

Global Schedule

- Let $T = \{T_1, \cdots, T_n\}$ be a set of distributed transactions executed over m sites with local schedules $\{S_1, \cdots, S_m\}$
- A schedule S is a global schedule for T and $\{S_1, \cdots, S_m\}$ if each S_i is a subsequence of S

Example

- $T = \{T_1, T_2\}$, Site A = $\{x\}$, Site B = $\{y\}$
- $T_1 = R_1(x), W_1(x), R_1(y), W_1(y)$
- $T_2 = R_2(x), R_2(y)$
- Local schedules:
  - S_A: $R_1(x), W_1(x), R_2(x)$
  - S_B: $R_2(y), R_1(y), W_1(y)$
- Which of the following schedules is a global schedule for T & $\{S_A, S_B\}$?

S_1: $R_1(x), R_2(x), R_2(y), W_1(x), R_1(y), W_1(y)$

S_2: $R_1(x), W_1(x), R_2(x), R_2(y), R_1(y), W_1(y)$

Serializable Global Schedule

- Let T be a set of transactions executed over m sites with local schedules $\{S_1, \cdots, S_m\}$
- **Theorem 5**: A global schedule S for T and $\{S_1, \cdots, S_m\}$ is view/conflict serializable if
    1. Each local schedule S_i is view/conflict serializable, and
        - ★ for each S_i, there exists a serial schedule S_i' that is view/conflict equivalent to S_i
    2. The local serialization orders are compatible
        - ★ there exists a serial schedule S' over T such that each S_i' is a subsequence of S'

Example

- $T = \{T_1, T_2\}$, Site A = $\{x\}$, Site B = $\{y\}$
- $T_1 = R_1(x), W_1(x), R_1(y), W_1(y)$
- $T_2 = R_2(x), R_2(y)$
- Local schedules:
  - S_A: $R_1(x), W_1(x), R_2(x)$
  - S_B: $R_2(y), R_1(y), W_1(y)$
- Is there a serializable global schedule for T and $\{S_A, S_B\}$?

Concurrency Control Protocols

- Lock-based protocols
- Timestamp-based protocols
- Optimistic protocols
- Mutiversion protocols
- Hybrid protocols

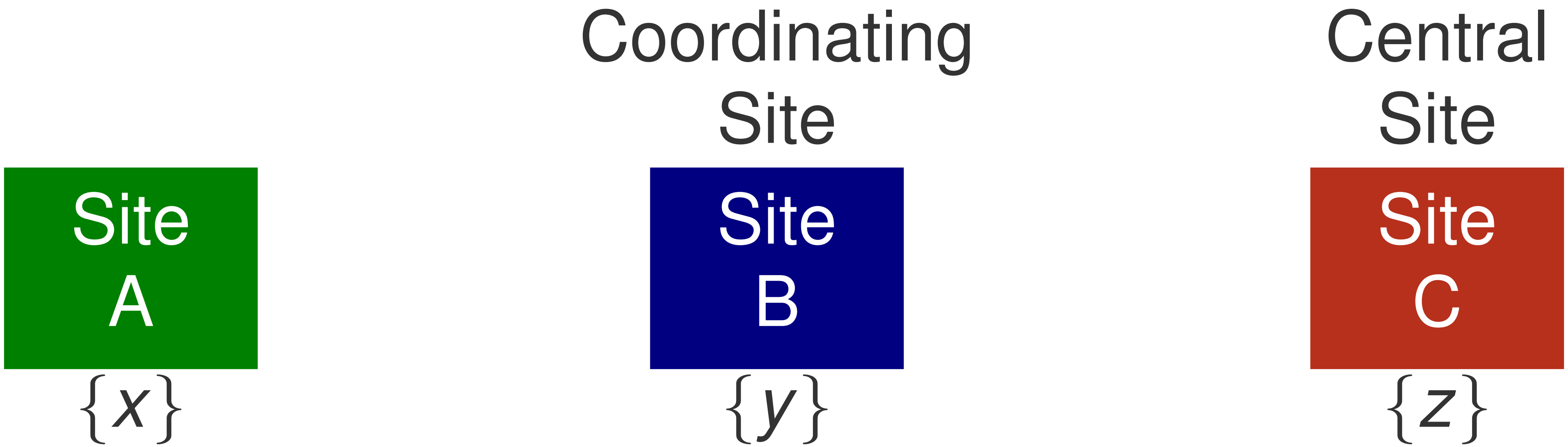
Distributed Lock-based Protocols

- Centralized 2PL (C2PL)
- Distributed 2PL (D2PL)

Centralized 2PL (C2PL)

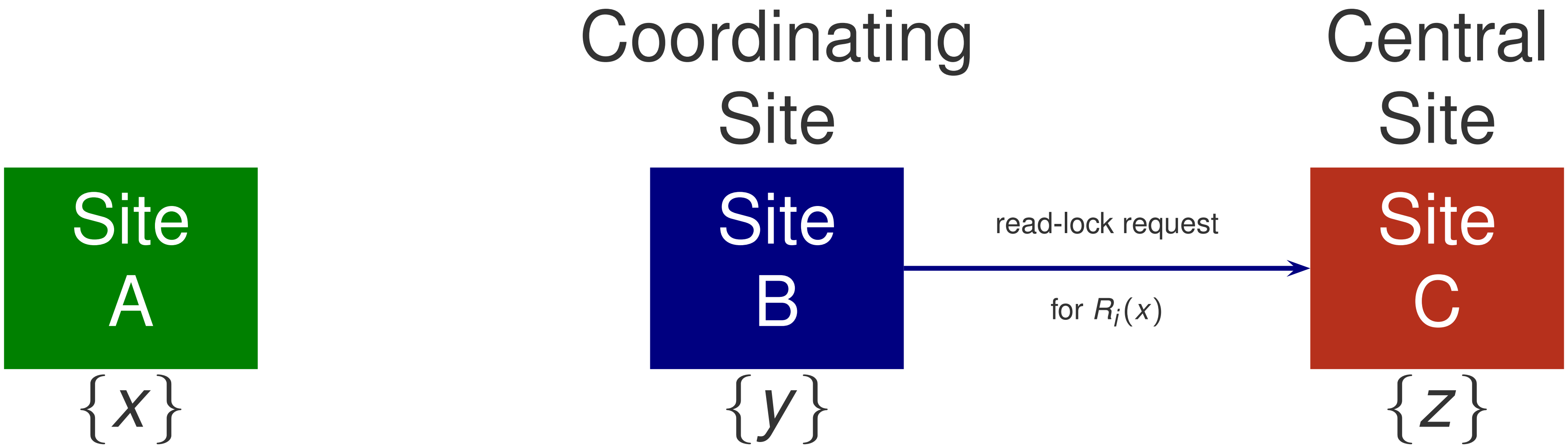
- One site is designated as central site
  - Locks are managed by only the central TM's lock manager
- Coordinating TM makes lock requests/releases to central TM

Centralized 2PL (C2PL): Example



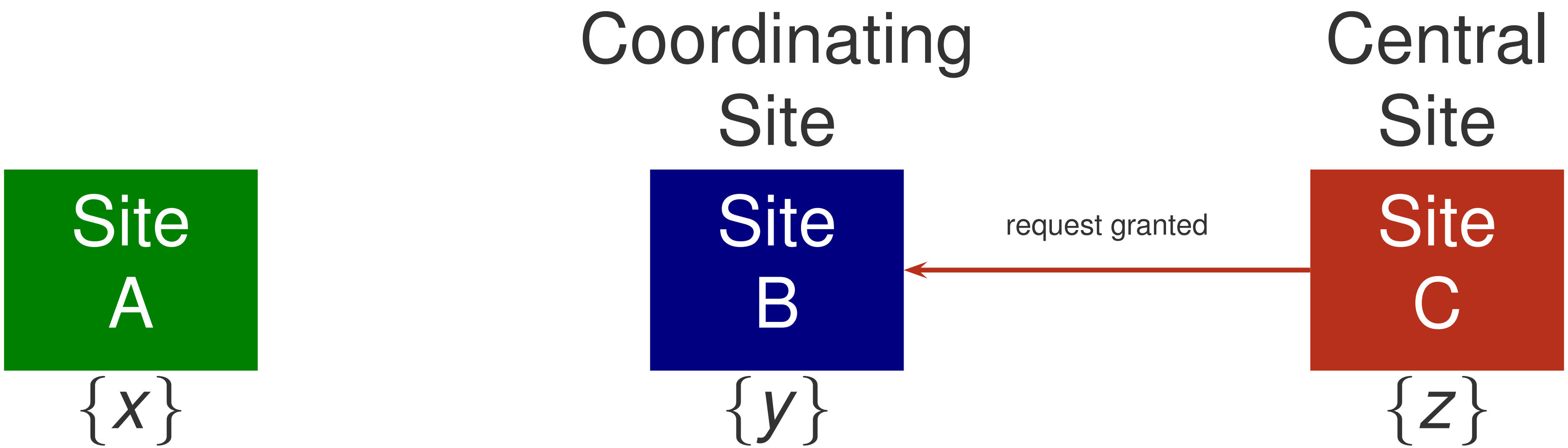
Site B is the coordinating site for a Xact T_i that needs to read x

Centralized 2PL (C2PL): Example



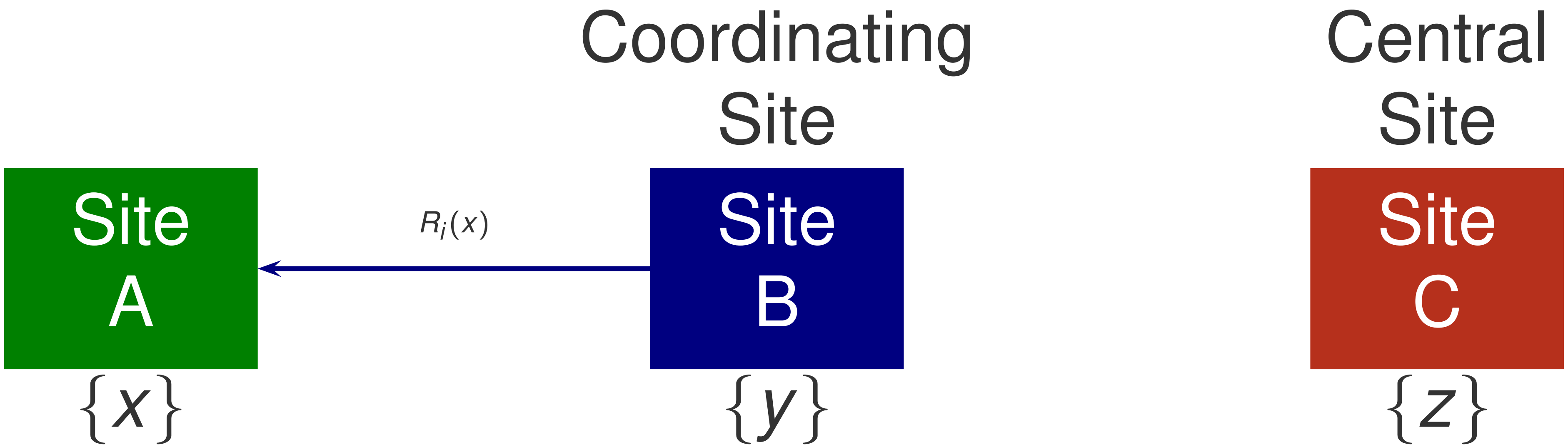
Coordinating TM sends a read-lock request for x to central TM

Centralized 2PL (C2PL): Example



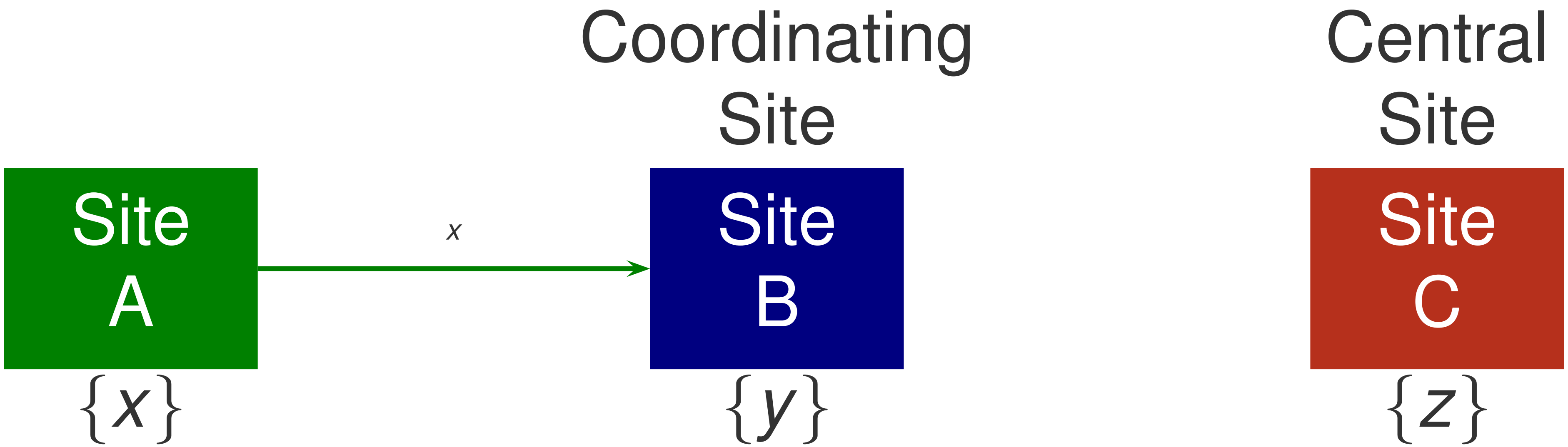
Central TM grants T_i's read-lock request on x & sends acknowlegement to coordinating TM

Centralized 2PL (C2PL): Example



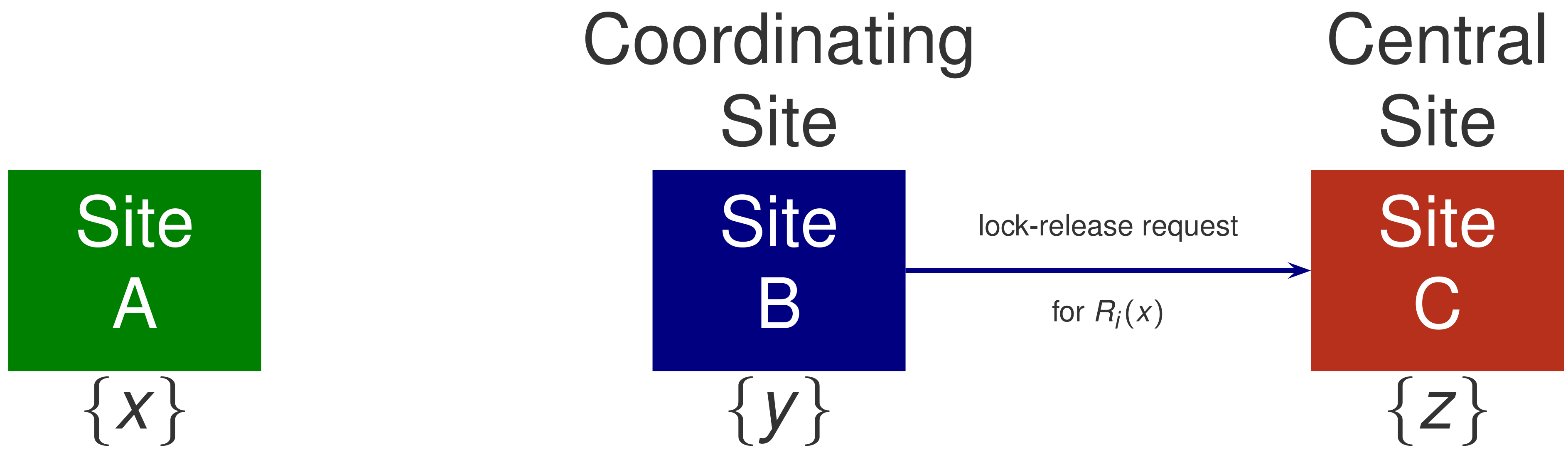
Coordinating TM sends Read$_i$(x) to Site A's TM

Centralized 2PL (C2PL): Example



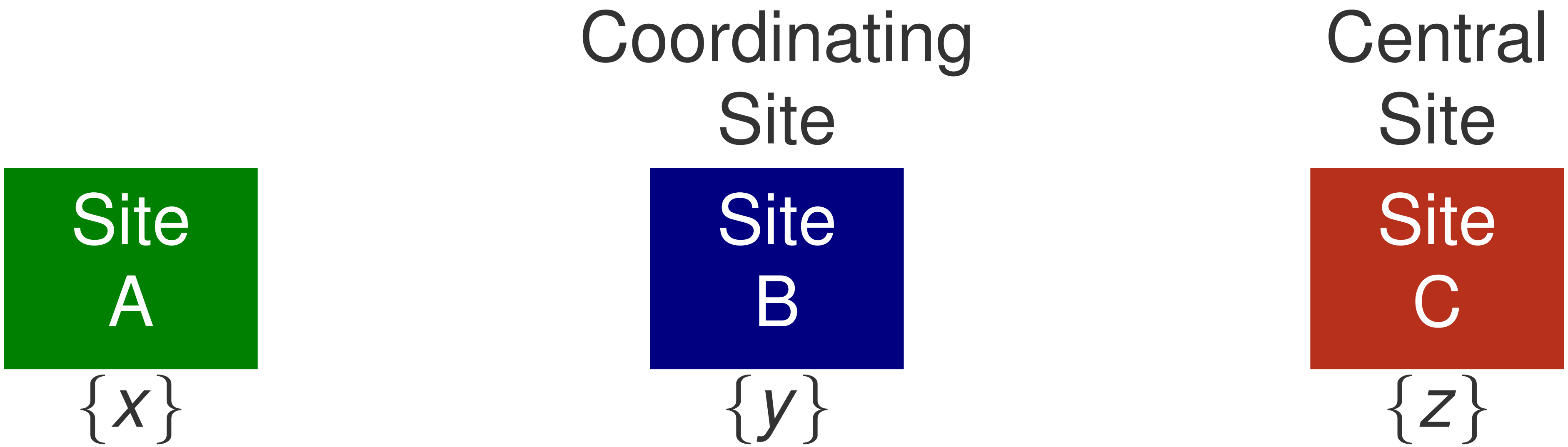
Site A's TM sends x to coordinating TM

Centralized 2PL (C2PL): Example



When T_i is done, coordinating TM sends a lock release notification for x to central TM

Centralized 2PL (C2PL): Example

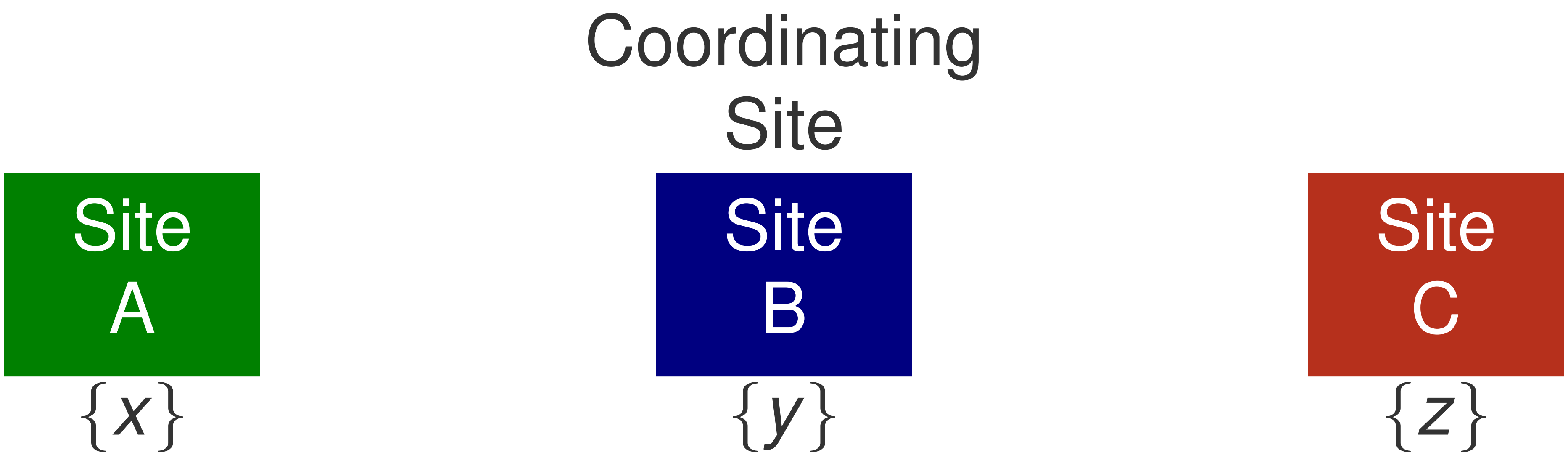


Central TM releases T_i's lock on x

Distributed 2PL (D2PL)

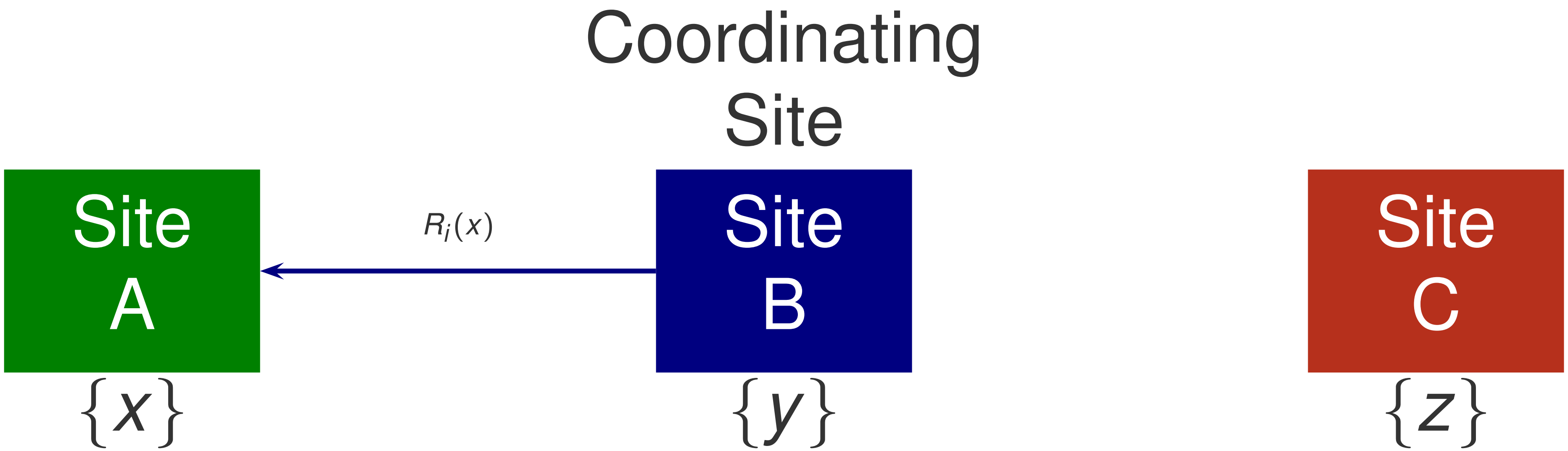
- Locks are managed collectively by each site's lock manager
- Early users: IBM's System R* & Tandem's NonStop SQL

Distributed 2PL (D2PL): Example



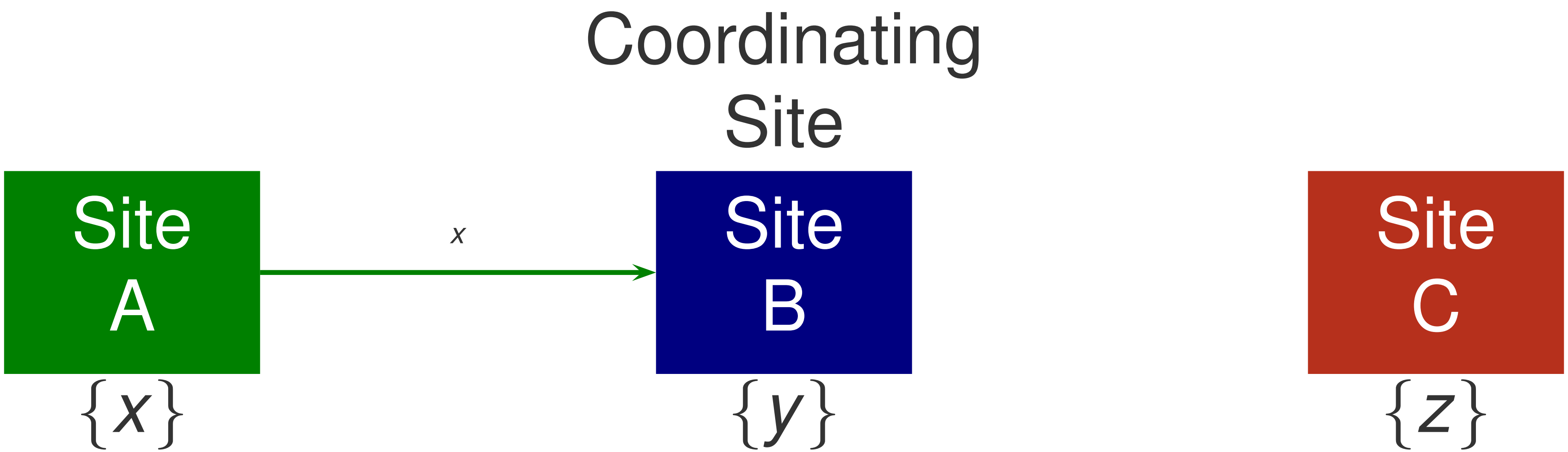
Site B is the coordinating site for a Xact T_i that needs to read x

Distributed 2PL (D2PL): Example



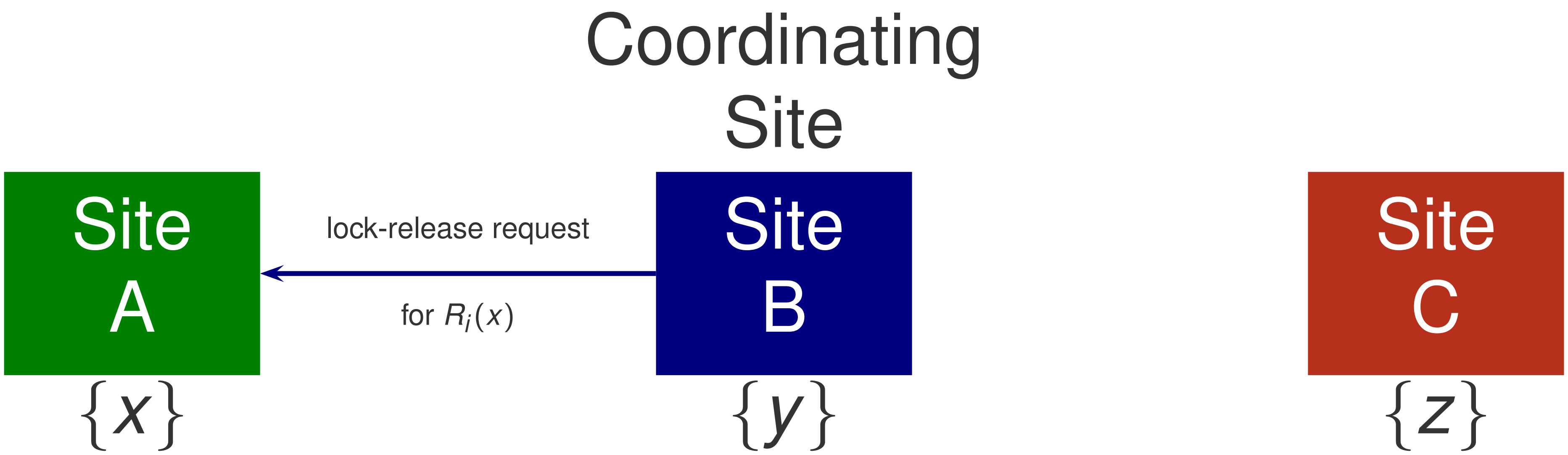
Coordinating TM sends $\text{Read}_i(x)$ to Site A's TM

Distributed 2PL (D2PL): Example



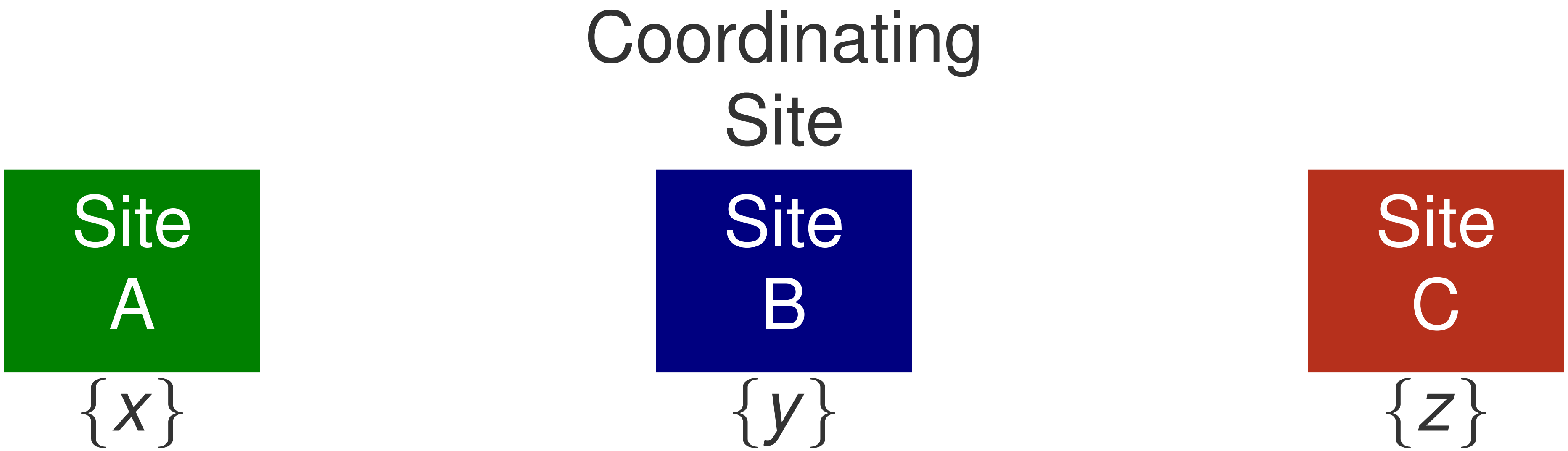
Site A's TM grants T_i's read-lock request on x & sends x to coordinating TM

Distributed 2PL (D2PL): Example



When T_i is done, coordinating TM sends a lock release request for x to Site A's TM

Distributed 2PL (D2PL): Example



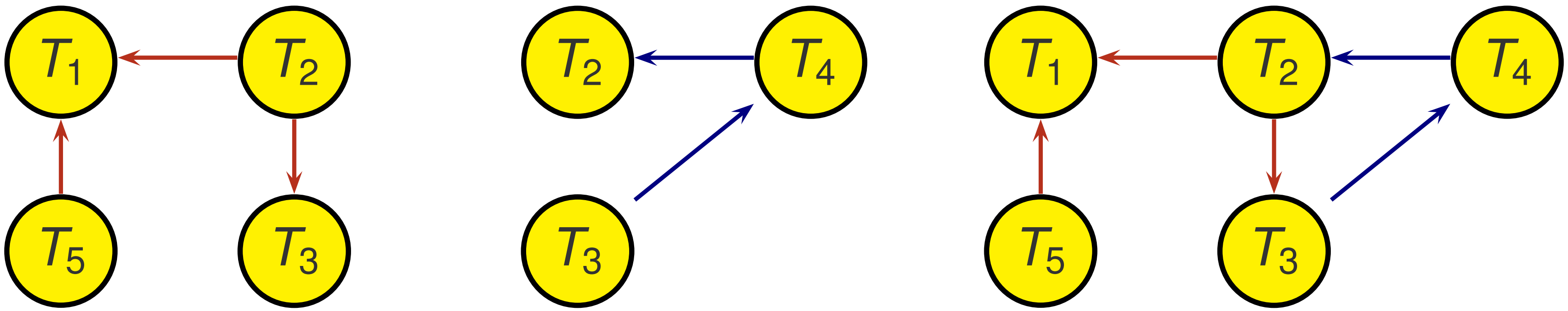
Site A's TM releases T_i's lock on x

Distributed Deadlock Detection

- Centralized approach
- Distributed approaches
  - Edge Chasing Algorithm
  - etc.

Centralized Approach

- One site is designated deadlock detector
- Each site maintains a Local Wait-For Graph (LWFG)
- Periodically, each site transmits its LWFG to the deadlock detector. Deadlock detector constructs a Global Wait-For Graph (GWFG) & looks for cycles in it
- Need more than one LWFG to detect distributed deadlocks



LWFG at Site 1 **LWFG at Site 2** **GWFG**

Distributed Snapshot Isolation Protocols

- **Key Challenge**: How to synchronize timestamps in distributed environment?
- **Protocols**
  - Centralized Snapshot Isolation (CSI)
  - Distributed Snapshot Isolation (DSI)

Centralized Snapshot Isolation (CSI)

- One site is designated as Centralized Coordinator (CC)
  - Responsible for assigning **start & commit timestamps**
- start(T_i): start timestamp of T_i
- commit(T_i): commit timestamp of T_i
- lastCommit(T_i) = commit(T_j), where T_j is the last Xact that committed before start(T_i)
- lastCommit(T_i) $<$ start(T_i) $<$ commit(T_i)
- Assume using FUW Rule to enforce concurrent update property
- Write locks are managed collectively by each site's lock manager

Centralized Snapshot Isolation (CSI)

- Start a new Xact T_i
  - TC sends a request to CC to obtain start(T_i) & lastCommit(T_i)
- $R_i(x)$ where x is stored at site A
  - TC sends read request & lastCommit(T_i) to TM_A
  - TM_A sends the most recent version of x w.r.t. lastCommit(T_i) to TC
- $W_i(x)$ where x is stored at site A
  - TC sends write request to TM_A
  - TM_A checks if write-lock on x can be granted to T_i
    - If granted, TM_A performs the update & sends notification to TC; otherwise, T_i is blocked
- When a Xact T commits and releases its locks, all Xacts that are blocked by T are aborted

Centralized Snapshot Isolation (CSI)

- Commit a Xact T_i
  - TC sends a request to CC to obtain commit(T_i)
  - TC executes the following modified variant of 2PC Protocol:
    - ★ In voting phase, TC includes **start(T_i) & commit(T_i)** in **PREPARE messages**
    - ★ When a participant P receives a PREPARE message, P checks if there are any WW-conflicts between T_i & all committed concurrent Xacts. If there exists some object x that is updated by T_i and there exists a version of x created by a Xact T_j where start(T_i) $<$ commit(T_j) $<$ commit(T_i), then P will vote to abort T_i

Centralized SI: Example

- Centralized Coordinator: Server C
  - Last commit timestamp = 600

Active Xact T	TC	last Commit(T)	start(T)
T_1	D	600	620
T_2	B	600	640

Server	Storage	X-Lock Status
A	x_{100}, x_{300}	(x, T_1, null)
B	y_{100}, y_{300}	(y, T_1, null)
C	z_{300}, z_{500}	(z, T_2, null)
D	v_{600}	null

- Each version of object O is denoted by O_t
  - t is the commit timestamp of the Xact that created O_t
- Each X-lock status entry is of the form (O, T, L)
  - O denote the object being locked
  - T denote the Xact holding a X-lock on O
  - L denote the list of Xacts being blocked by T for object O

Centralized SI: Example (cont.)

- Centralized Coordinator: Server C
  - Last commit timestamp = 600

Active Xact T	TC	last Commit(T)	start(T)
T_1	D	600	620
T_2	B	600	640

Server	Storage	X-Lock Status
A	x_{100}, x_{300}	(x, T_1, null)
B	y_{100}, y_{300}	(y, T_1, null)
C	z_{300}, z_{500}	(z, T_2, null)
D	v_{600}	null

- T_1 decides to commit
  - Server D requests Server C for T_1's commit timestamp
  - Server C replies with 650
  - Server D sends PREPARE message to Servers A & B
    - PREPARE message contains (start(T_1), commit(T_1)) = (620, 650)
  - Both Servers A & B reply with VOTE-COMMIT
  - Server D commits T_1 following 2PC protocol

Centralized SI: Example (cont.)

- Centralized Coordinator: Server C
  - Last commit timestamp = 650

Active Xact T	TC	last Commit(T)	start(T)
T_2	B	600	640

Server	Storage	X-Lock Status
A	$x_{100}, x_{300}, x_{650}$	null
B	$y_{100}, y_{300}, y_{650}$	null
C	z_{300}, z_{500}	(z, T_2, null)
D	v_{600}	null

- T_2 updates x
  - Server B sends T_2's write request to Server A
  - Server A grants X-lock for T_2's write request
    - Server A's X-lock status entry: (x, T_2, null)
- T_2 decides to commit
  - Server B requests Server C for T_2's commit timestamp
  - Server C replies with 660
  - Server B sends PREPARE message to Servers A & C
    - PREPARE message contains (start(T_2), commit(T_2)) = (640, 660)
  - Server A replies with VOTE-ABORT
  - Server C replies with VOTE-COMMIT
  - Server B aborts T_2 following 2PC protocol

References

- T. Özsu & P. Valdureiz, *Distributed Transaction Processing*, Chapter 5, Principles of Distributed Database Systems, 4^{th} Edition, 2020