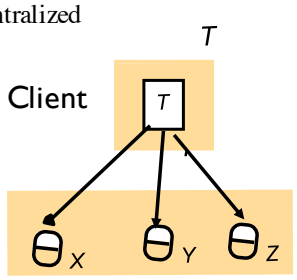
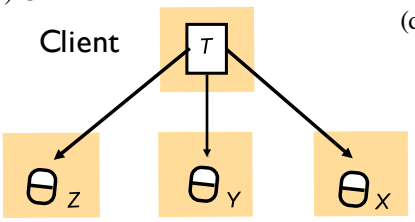
Distributed Transactions

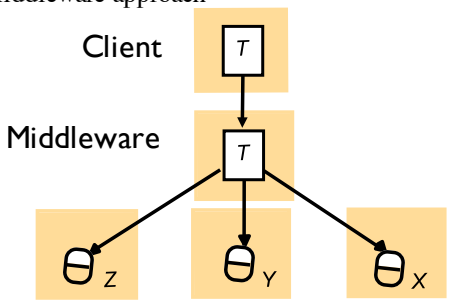
(a) Centralized



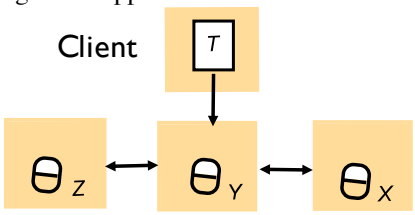
(b) Client-Aware Distribution



(c) Middleware approach

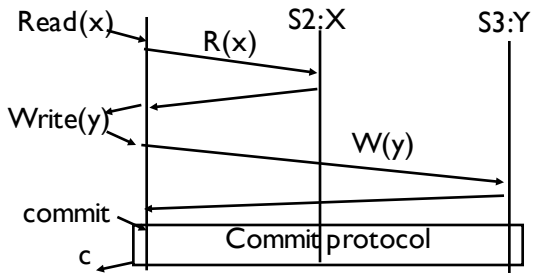


(d) Homogenous Approach



Distributed Transaction Processing

- Collection of sites S^i, i= 1, …. N
- Each process is responsible for a (finite) set D^i of data items;
- We first assume no replication, that is $D^i \cap D^j = \{\}$ for i,j = 1, …. N, $i \neq j$
- Homogenous: each server knows the location of all data items
- Middleware: middleware knows the location of all data items
- Transaction T_i is submitted to one site S^j, called the coordinator of T_i (or to the global transaction manager / middleware)
  - For each operation $O_{ik}(x)$ of transaction T_i, $O_{ik}(x)$ is executed at site S^l where $x \in D^l$, S^l sends an ack to S^j
  - At the end of T_i , all participating sites terminate (a,c) T_i



Global and local Histories

- Distributed transaction processing produces local histories at each site
  - ☆ The local history at site S_j contains for each transaction T_i the operations that have been executed at S_j.
- The global history is the union of all local histories.
- A concurrency control mechanism guarantees global serializability if all global histories that might occur are conflict-serializable

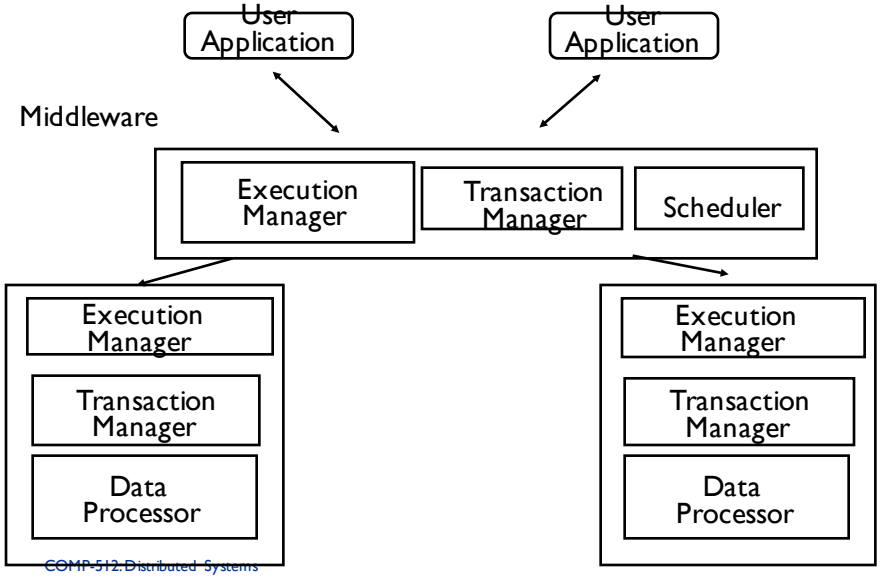
Independent Execution

T1	T2
r1(x)	
	r2(z)
	r2(y)
w1(x)	
w1(y)	
c1	
	w2(x)
	c2

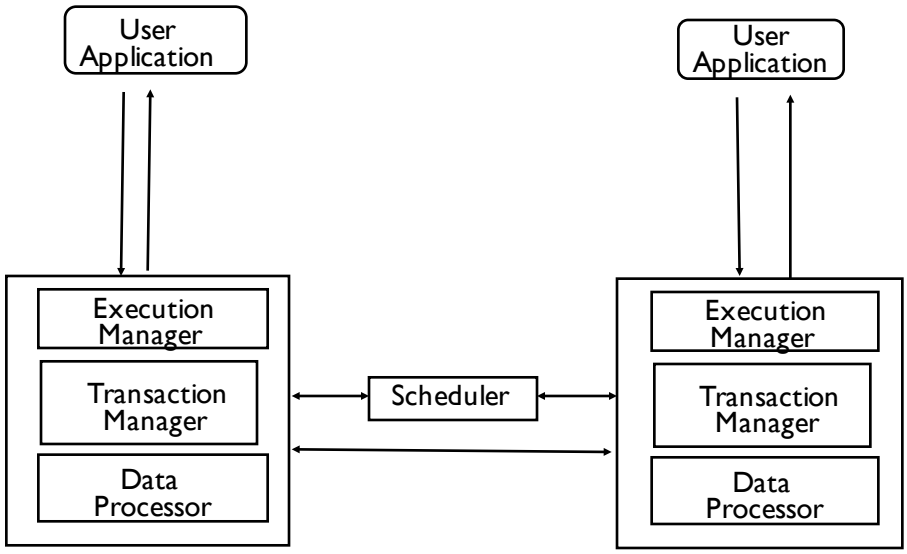
S1:x	S2:y	S3:z
r1(x)		
		r2(z)
	r2(y)	
w1(x)		
	w1(y)	
C1	c1	
w2(x)		
C2	c2	c2

- Local histories are serializable, but the global history is not serializable

Centralized Scheduler Middleware-based System

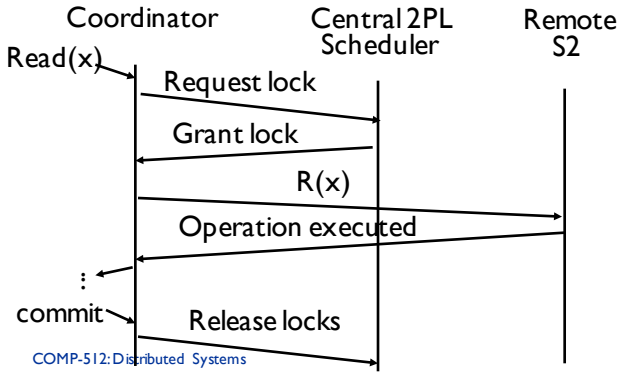


Centralized Scheduler Homogenous System

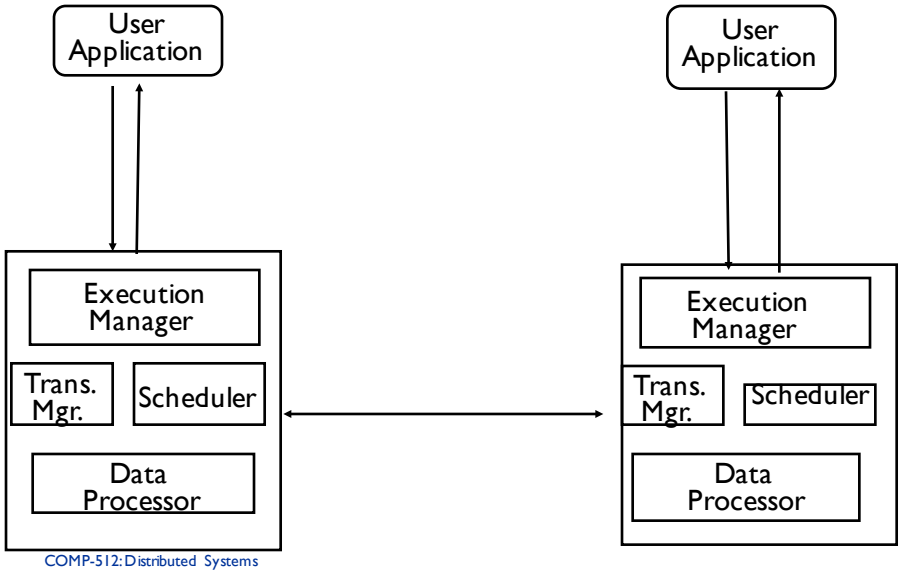


Centralized 2PL

- There exists one global 2PL Scheduler
- Let S1 be the server to which transaction T_i was submitted:
  - For each operation $o_{ij}(X)$, TM1 first requests the corresponding lock from the central 2PL scheduler
  - Once the lock is granted, the operation is forwarded to the server S2 maintaining X.
  - In a middleware based system, global scheduler is typically part of the middleware

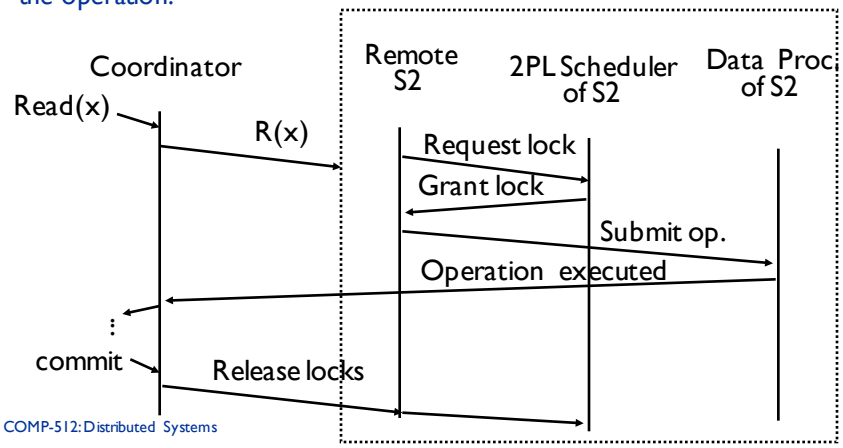


Distributed Scheduler homogenous system



Distributed 2PL

- Each server has its own local 2PL scheduler
- Let S1 be the server to which transaction T_i was submitted:
  - For each operation $o_{ij}(X)$, TM1 forwards the operation to the TM2 of server S2 maintaining X
  - The remote site first acquires a lock on X and then submits the execution of the operation.



Deadlock in distributed system

- Timeout Mechanism
  - if a transaction waits for a lock longer than a predefined timeout interval, assume it is in a deadlock cycle and abort the transaction
  - Disadvantage: choice of adequate timeout interval is crucial
- Global Deadlock Detection
  - Every site sends its local graph periodically to a central site
  - Central site will eventually detect conflict
  - Suitable with centralized 2PL
- Deadlock Prevention
  - Lock all items at the beginning of transaction
- Distributed Deadlock detection
  - seldomly used

Deadlock Detection (Continued)

Example:

T1: S(A), r(A), S(B)
T2: X(B),W(B) X(C)
T3: S(C), R(C) **X(A)**
T4: X(B)