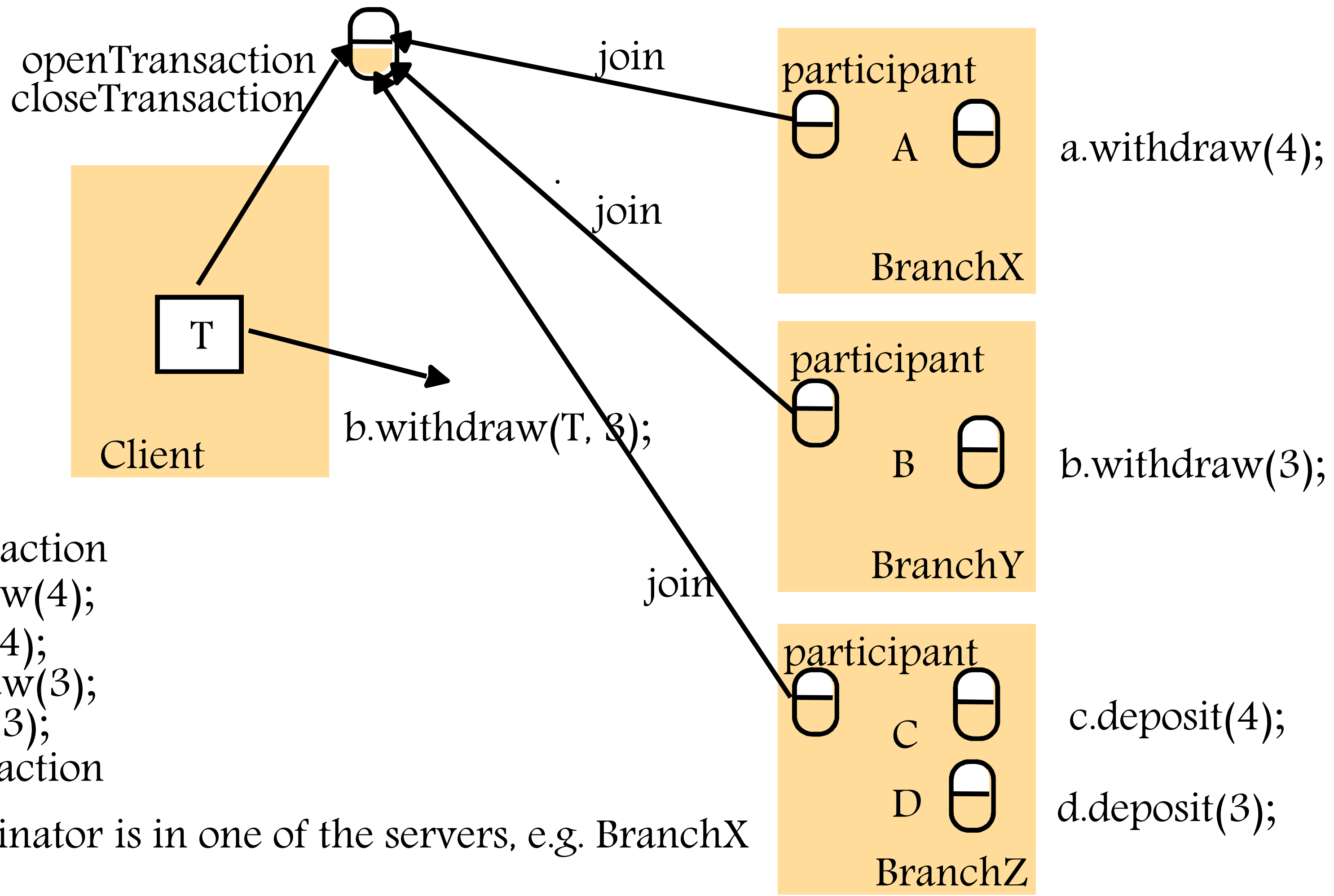
Distributed Deadlock Detection

George Coulouris, Jean Dollimore and Tim Kindberg, "Distributed Systems Concepts and Design", Fifth Edition, Pearson Education, 2012

Deadlock Detection

- Centralized Detection
- Distributed Detection
- Hierarchical Detection

A distributed banking transaction



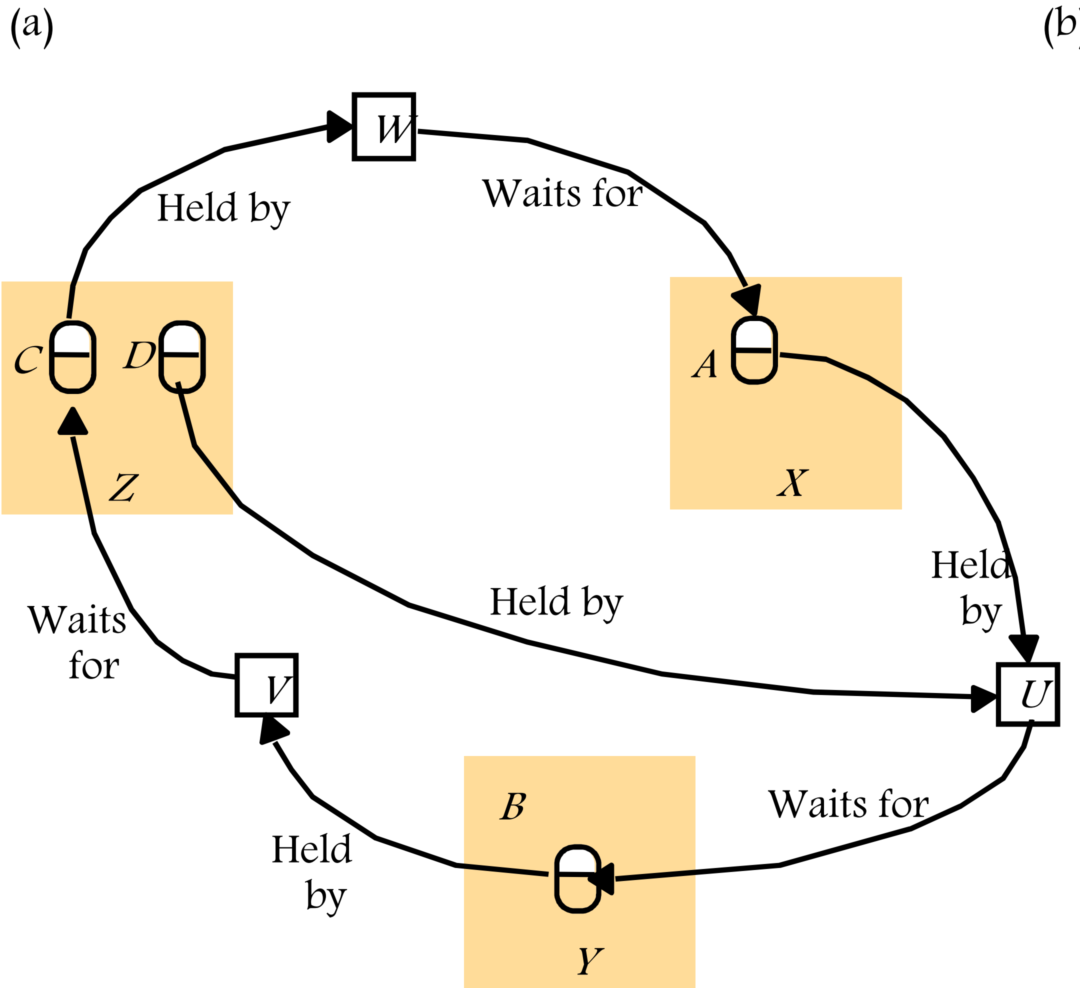
T = openTransaction
a.withdraw(4);
c.deposit(4);
b.withdraw(3);
d.deposit(3);
closeTransaction

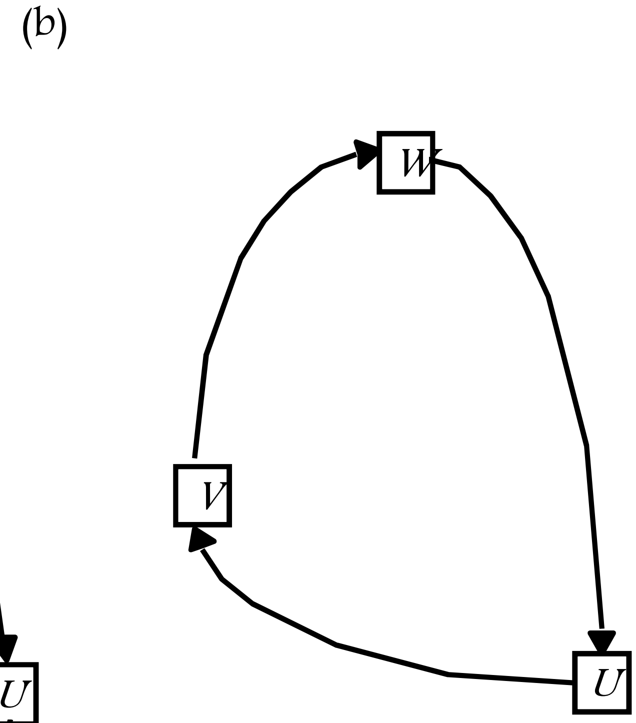
Note: the coordinator is in one of the servers, e.g. BranchX

Interleaving of transactions *U*, *V* and *W*

U		*V*		*W*	
d.deposit(10)	lock *D*				
		b.deposit(10)	lock *B* at *Y*		
a.deposit(20)	lock *A* at *X*				
				c.deposit(30)	lock *C* at *Z*
b.withdraw(30)	wait at *Y*				
		c.withdraw(20)	wait at *Z*		
				a.withdraw(20)	wait at *X*

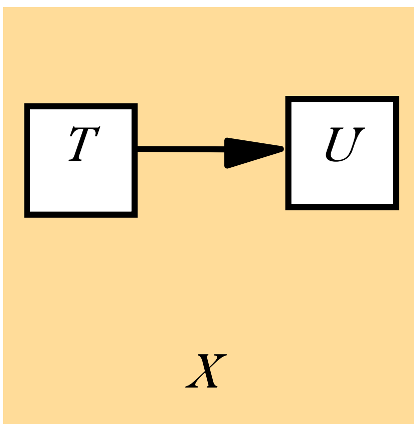
Distributed deadlock – Path-Pushing Method Obermarck's Algorithm



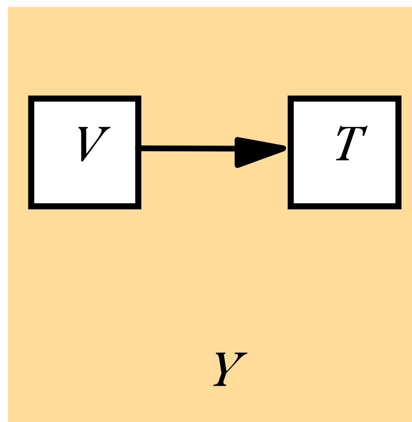


Local and global wait-for graphs

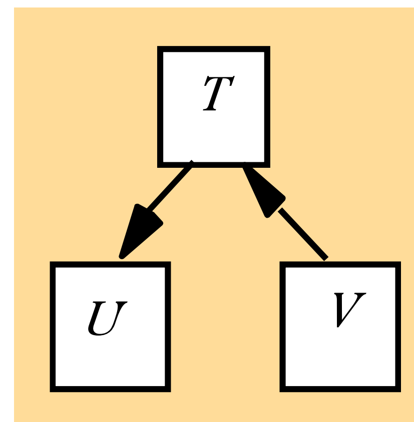
local wait-for graph



local wait-for graph



global deadlock detector



Distributed Deadlock Detection- Edge-Chasing Method – Chandy, Misra Haas's Algorithm

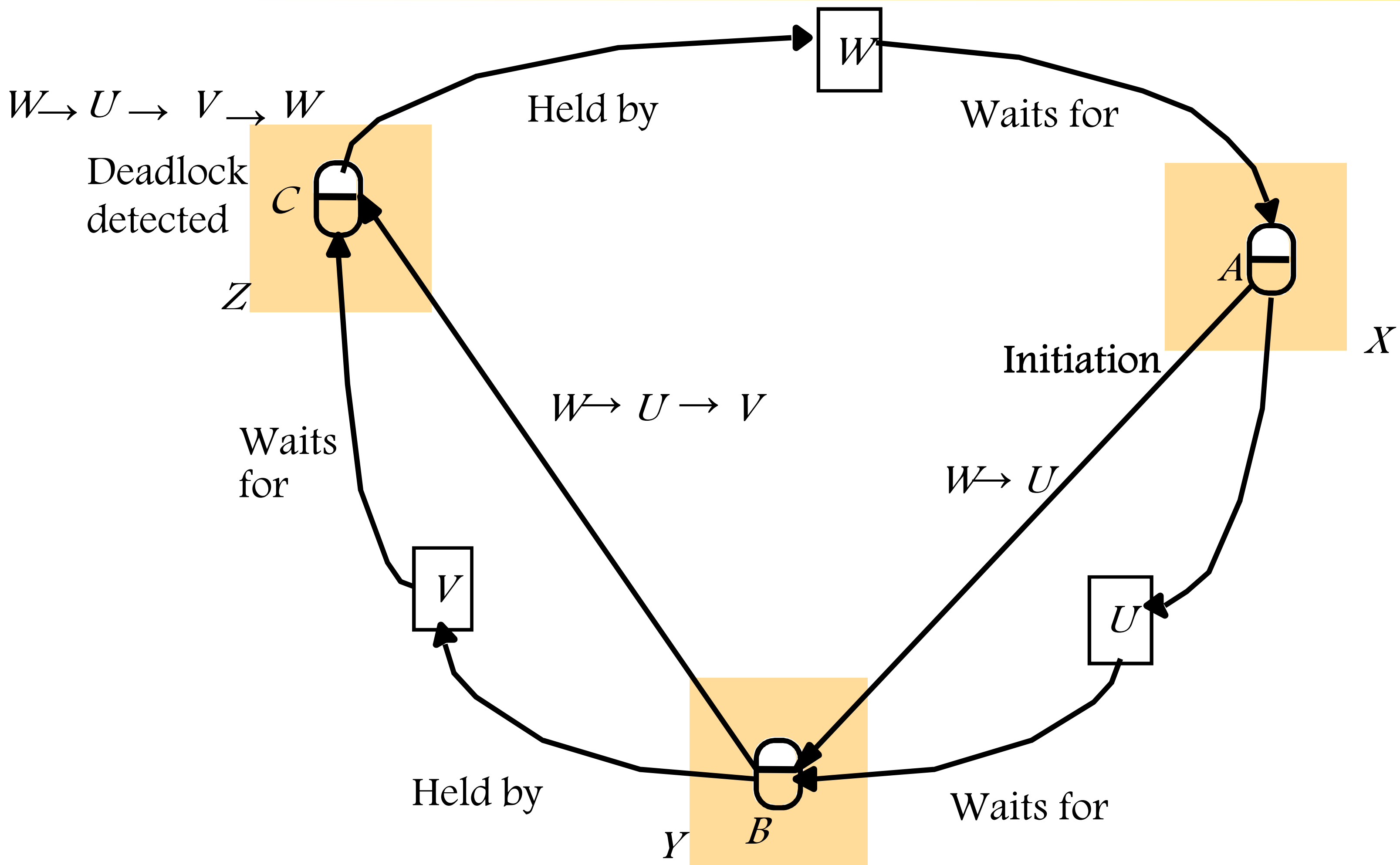
- Global wait-for graph is not constructed, but each of the servers involved has knowledge about some of its edges.
- The servers attempt to find cycles by forwarding messages called probes, which follow the edges of the graph throughout the distributed system.
- A probe message consists of transaction wait-for relationships representing a path in the global wait-for graph.
- The coordinator is responsible for recording whether the transaction is active or is waiting for a particular object, and participants can get this information from their coordinator.
- Lock managers inform coordinators when transactions start waiting for objects and when transactions acquire objects and become active again.
- When a transaction is aborted to break a deadlock, its coordinator will inform the participants and all of its locks will be removed

Distributed Deadlock Detection- Edge-Chasing Method – Chandy, Misra Haas's Algorithm

- 3 phases in Edge-Chasing Algorithm

1. Initialization of Deadlock Detection
2. Detection of Deadlock
3. Resolution of Deadlock

Probes transmitted to detect deadlock



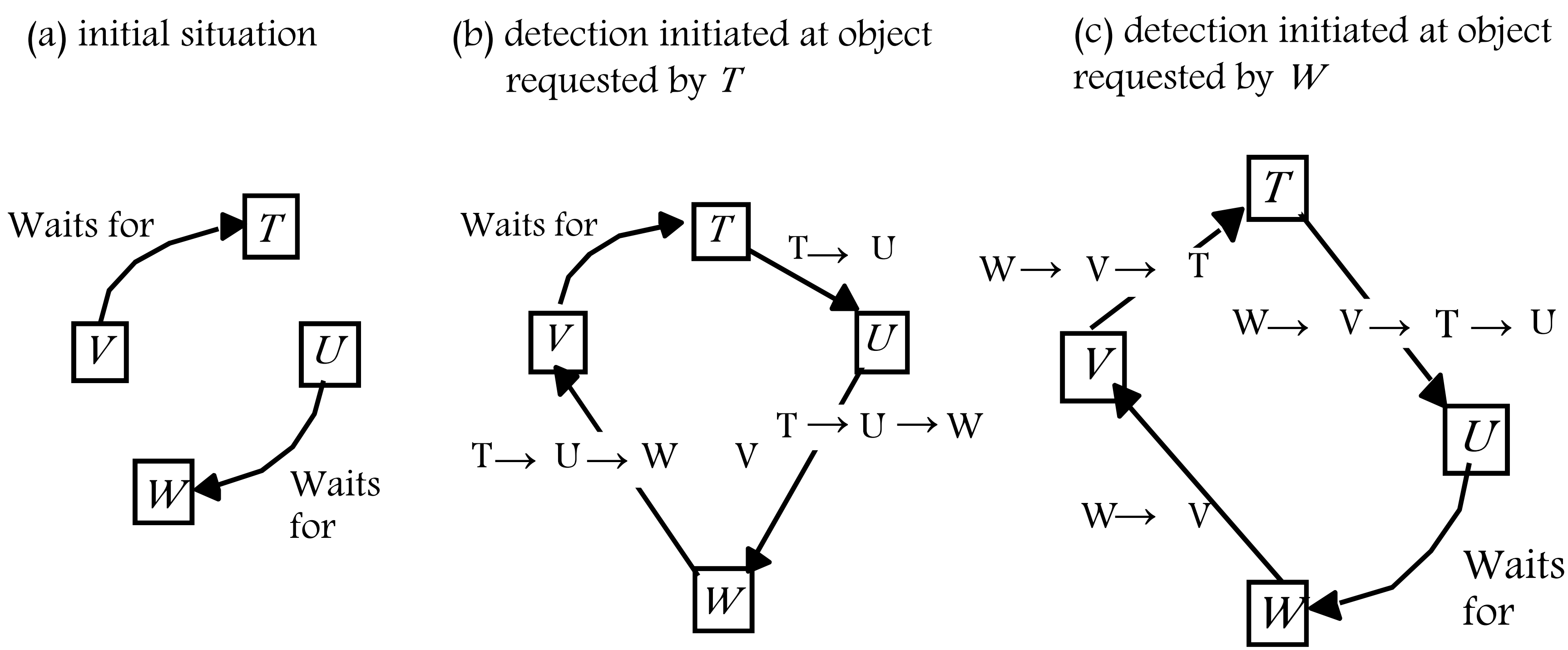
Distributed Deadlock Detection- Edge-Chasing Method – Chandy, Misra Haas's Algorithm

- Server X initiates detection by sending probe < W → U > to the server of B (Server Y).
- Server Y receives probe < W → U >, notes that B is held by V and appends V to the probe to produce < W → U → V >. It notes that V is waiting for C at server Z.
- This probe is forwarded to server Z.
- Server Z receives probe < W → U → V >, notes C is held by W and appends W to the probe to produce < W → U → V → W >.
- This path contains a cycle. The server detects a deadlock.

Transaction Priorities

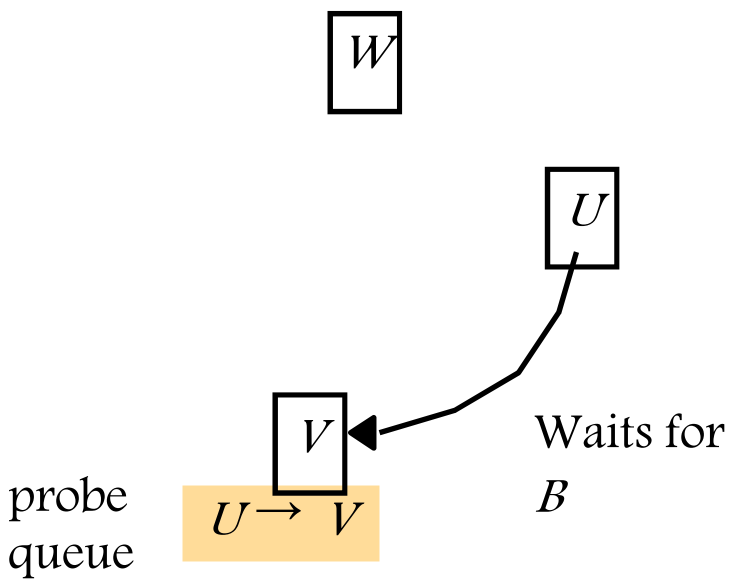
- Consider transactions T, U, V, W
- Priority is specified as T> U > V> W.
- T as highest Priority, W as least priority.
- Lowest Priority transaction will be aborted first.
- Highest Priority transaction will initiate the deadlock detection.
- Probe messages must be sent from higher priority transaction to lower priority transactions. – (Probe Downhill)
- Drawback: Deadlock may be left undetected.
- Solution: Probe Queue.

Two probes initiated

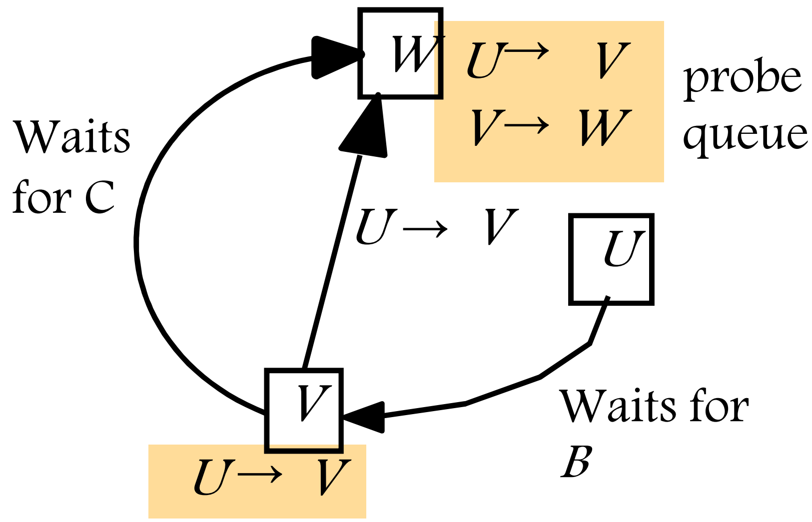


Probes travel downhill

(a) V stores probe when U starts waiting



(b) Probe is forwarded when V starts waiting



Summary

- Requires arrangements to pass on probes to new holders and to discard probes that refer to transactions that have been committed or aborted.
- If relevant probes are discarded, undetected deadlocks may occur, and if outdated probes are retained, false deadlocks may be detected.
- Kshemkalyani and Singhal [1994] argued that distributed deadlocks are not very well understood because there is no global state or time in a distributed system.