Capítulo 8
Transacciones distribuidas

Sistemas Distribuidos

Universidad Nacional de Asunción
Facultad Politécnica
Ingeniería Informática

Ing. Fernando Mancía

Transacciones distribuidas

Transacción plana o anidada que accede a objetos gestionados por múltiples servidores.

Involucran más de un servidor.

Cuando una transacción llega a su fin, la propiedad de atomicidad requiere que, todos los servidores completen su transacción o que todos aborten.

Uno de los servidores asume de coordinador que debe asegurar el mismo resultado en todos los servidores.

Figura 8.1
Transacciones distribuidas

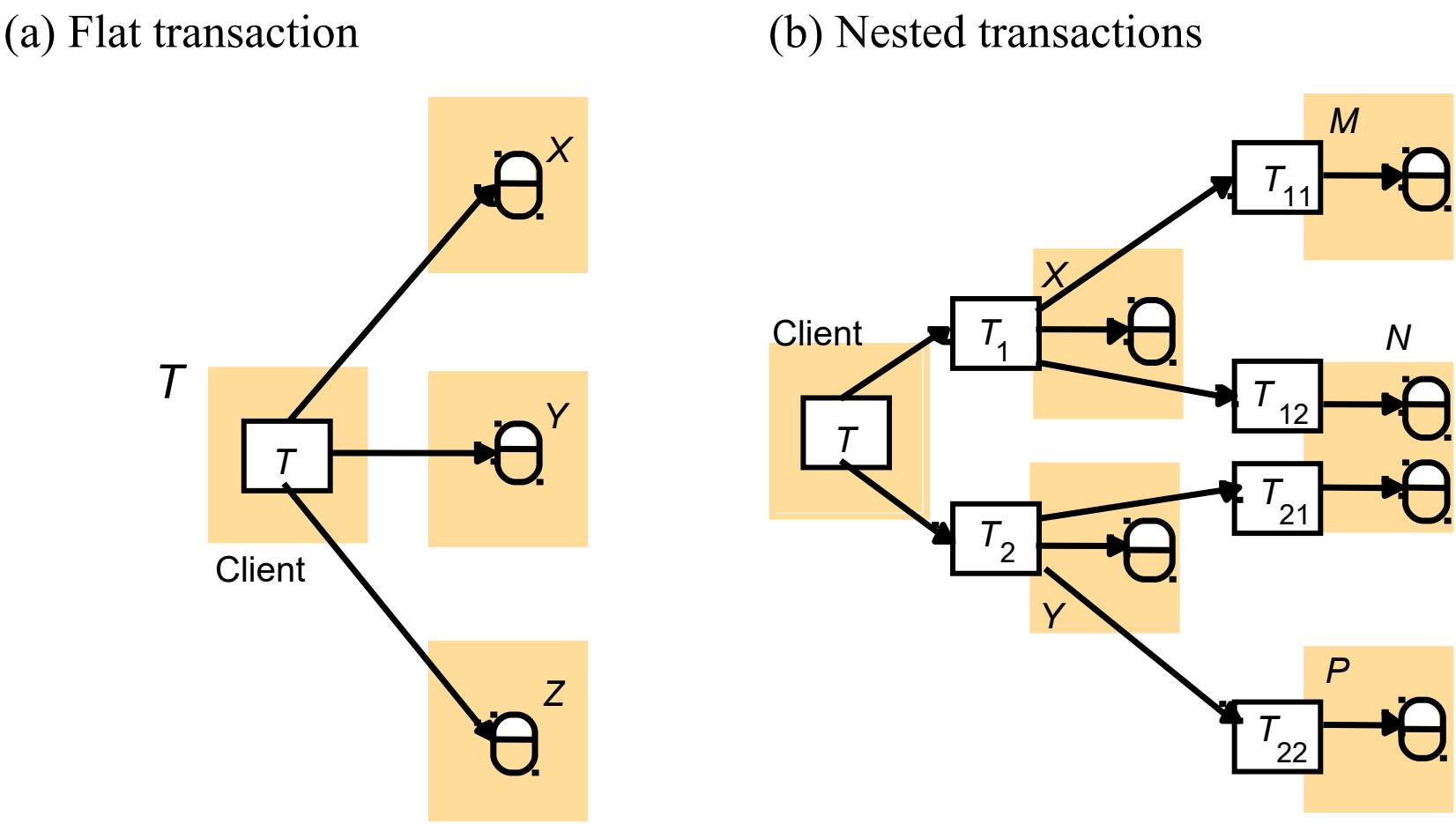
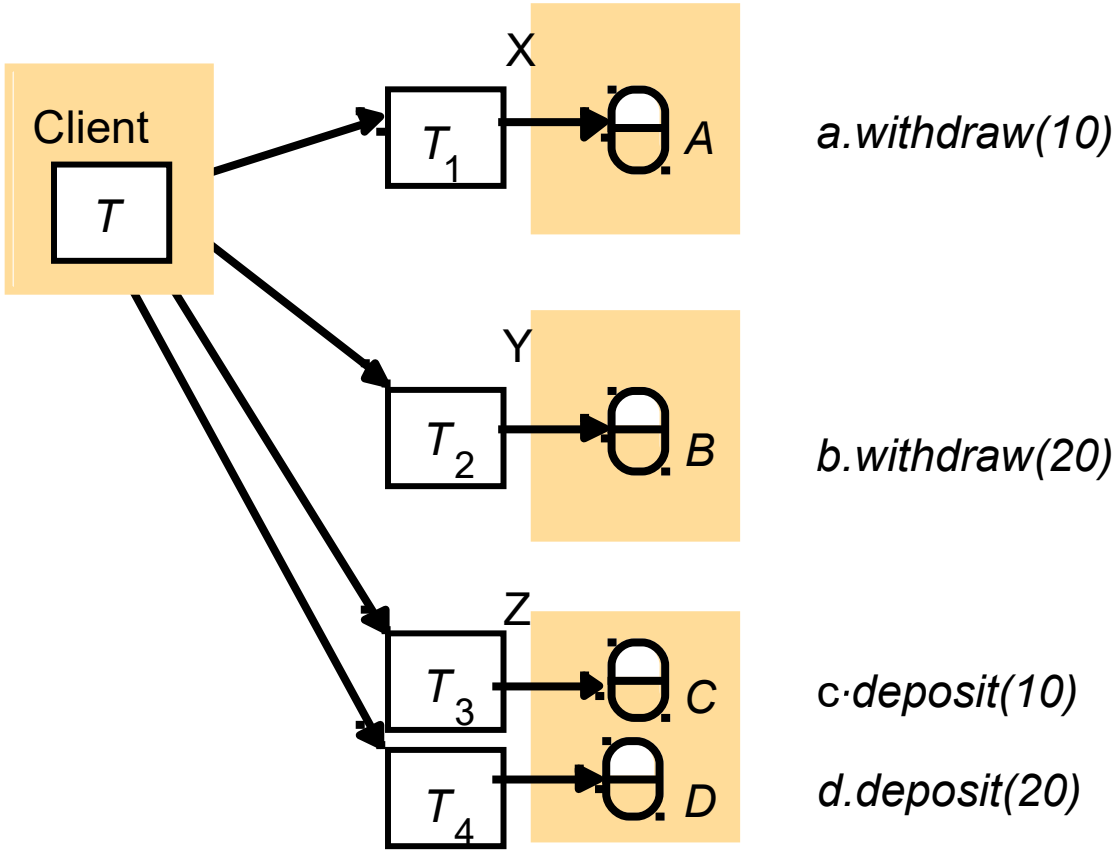


Figura 8.2
Transacción bancaria anidada

A -> C: 10
B -> D: 20



T = openTransaction

openSubTransaction
a.withdraw(10);

openSubTransaction
b.withdraw(20);

openSubTransaction
c.deposit(10);

openSubTransaction
d.deposit(20);

closeTransaction

El coordinador de una transacción distribuida

Los servidores que ejecutan peticiones como parte de una transacción distribuida necesitan poder comunicarse entre ellos para ***coordinar sus acciones*** cuando se consuma la transacción.

Un cliente comienza una transacción enviando una petición de *abreTransacción* al coordinador en cualquier servidor. El coordinador con el que se contacta lleva a cabo *abrirTransacción* y devuelve al cliente el identificador resultante de la transacción.

Los identificadores de transacciones para transacciones distribuidas han de ser únicos dentro del sistema distribuido. Una forma sencilla de obtener esto es que cada TID contenga dos partes: el identificador del servidor (por ejemplo una dirección IP) que la creó y un número único dentro del servidor.

El coordinador de una transacción distribuida

El coordinador que abrió la transacción se convierte en el *coordinador* para la transacción distribuida y, es el responsable final de consumarla o abortarla.

Participante. Cada uno de los servidores que gestione un objeto al que accede la transacción es un participante en la transacción y proporciona un objeto que llamaremos *participante.* Cada participante es responsable de seguir la pista de todos los objetos recuperables en el servidor implicado en la transacción.

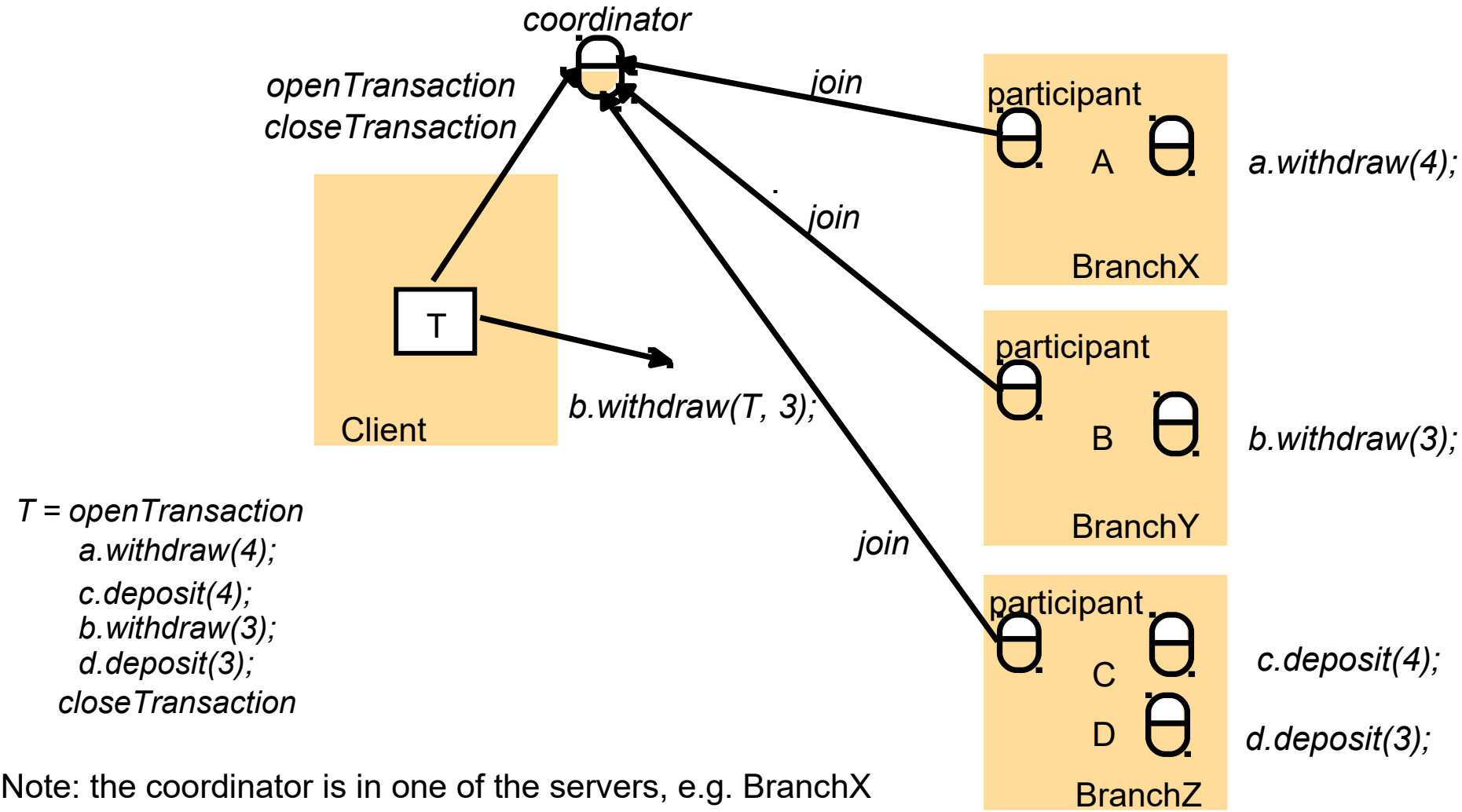
El coordinador de una transacción distribuida

Los participantes son responsables de cooperar con el coordinador para sacar adelante el protocolo de consumación.

Referencias. Durante el progreso de una transacción el coordinador registra una lista de referencias participantes, y cada participante registra una referencia hacia el coordinador.

El hecho de que el coordinador conozca a todos los participantes y que cada participante conozca al coordinador les permite recoger información necesaria al momento de la consumación.

Figura 8.3
Un ejemplo de transacción bancaria distribuida



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

Figura 8.4
Operaciones para el protocolo de consumación en dos fases

puedeConsumar?(trans) → *Sí / No* Llamada desde el coordinador al participante para preguntar si puede consumar una transacción. El participante responde con su voto.
Consuma(trans) Llamada desde el coordinador al participante para decirle que consume su parte de una transacción.
Aborta(trans) Llamada desde el coordinador al participante para decirle que aborte su parte de una transacción.
heConsumado(trans, participante) Llamada desde el participante al coordinador para confirmar que ha consumado la transacción.
dameDecisión(trans) → *Sí / No* Llamada desde el participante al coordinador para preguntar por la decisión sobre una transacción tras haber votado *Sí* aunque no ha obtenido respuesta tras cierto tiempo. Se utiliza para recuperarse de la caída de un servidor o de mensajes con retraso.

Figura 8.5
El protocolo de consumación en dos fases

Fase 1 (fase de votación):

1. El coordinador envía una petición *puedeConsumar?* a cada participante en la transacción.
2. Cuando un participante recibe una petición *puedeConsumar?*, responde al coordinador con su voto (*Sí* o *No*). Antes de votar *Sí*, se prepara para consumar, guardando los objetos en un dispositivo de almacenamiento permanente. Si el voto es *No*, el participante aborta inmediatamente.

Fase 2 (finalización en función del resultado de la votación)

3. El coordinador recoge los votos (incluyendo el propio).
   (a) Si no hay fallos y todos los votos son *Sí*, el coordinador decide consumar la transacción y envía peticiones de *Consuma* a cada uno de los participantes.
   (b) En otro caso, el coordinador decide abortar la transacción y envía peticiones *Aborta* a todos los participantes que votaron *Sí*.
4. Los participantes que han votado *Sí* están esperando por una petición *Consuma* o *Aborta* por parte del coordinador. Cuando un participante recibe uno de estos mensajes, actúa en función de ellos, y en el caso de *Consuma*, realiza una llamada de *heConsumado* como confirmación hacia el coordinador.

Figura 8.6
Comunicación en el protocolo de consumación en dos fases

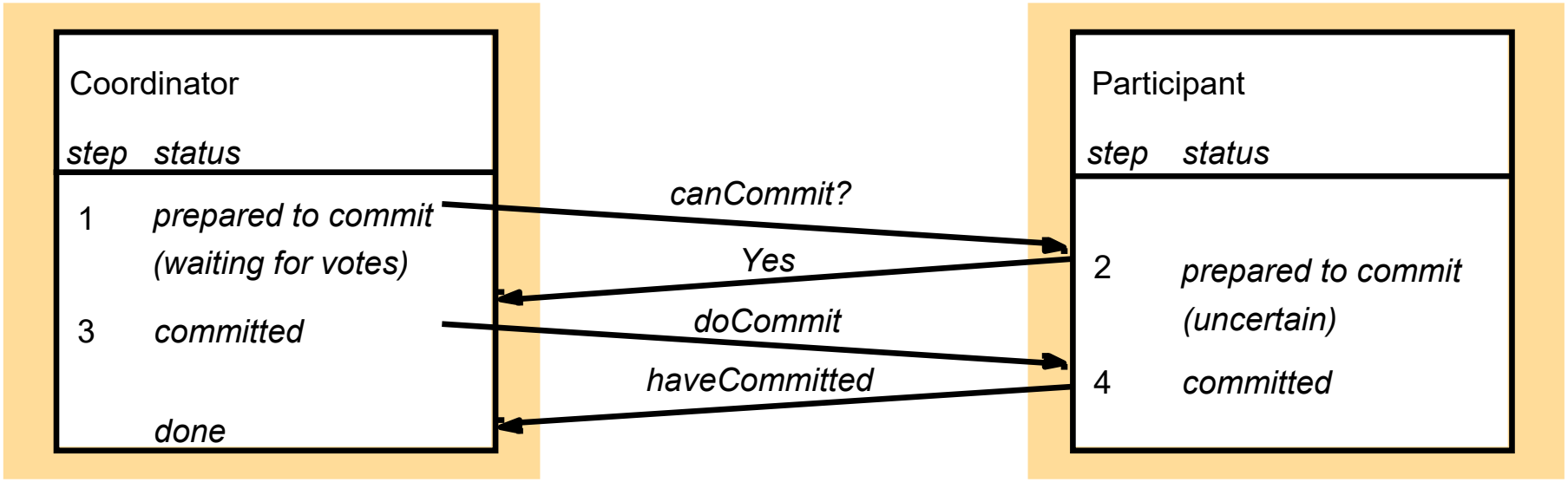


Figura 8.7
Operaciones en el coordinador para las transacciones anidadas

abreSubTransacción(trans) → *subTrans*

Abre una nueva subtransacción cuya madre es *trans*, y devuelve un identificador de subtransacción único.

dameEstado(trans) → *consumada, abortada, provisional*

Pide al coordinador que informe del estado de la transacción *trans*. Devuelve valores que representan uno de los siguientes: *consumada, abortada, provisional*.

La transacción T decide si se consuma

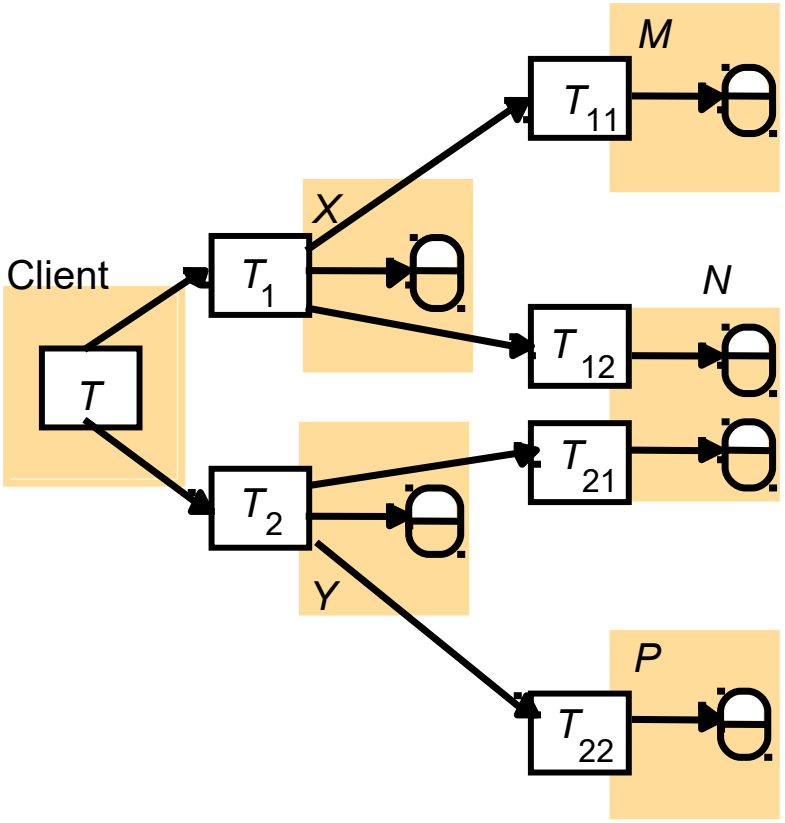


Figura 8.8
La transacción T decide si se consuma

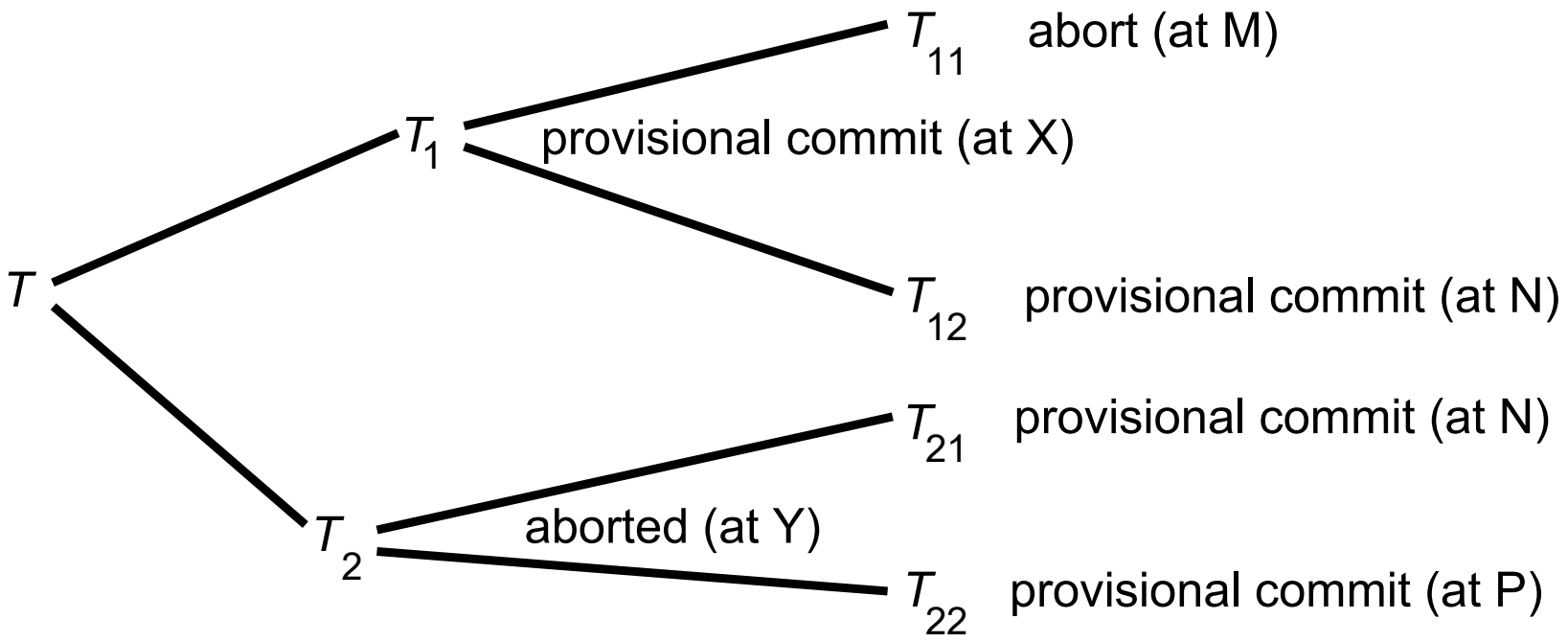


Figura 8.9
Información retenida por los coordinadores de transacciones anidadas

Coordinator of transaction	*Child transactions*	*Participant*	*Provisional commit list*	*Abort list*
T	T_1 , T_2	yes	T_1 , T_{12}	T_{11} , T_2
T_1	T_{11} , T_{12}	yes	T_1 , T_{12}	T_{11}
T_2	T_{21} , T_{22}	no (aborted)		T_2
T_{11}		no (aborted)		T_{11}
T_{12} , T_{21}		T_{12} but not T_{21}	T_{21} , T_{12}	
T_{22}		no (parent aborted)	T_{22}	

Figura 8.10
¿puede Consumar? Para el protocolo de dos fases jerárquico

puedeConsumar?(trans, subTrans) → *Sí / No*

Llama a un coordinador para que pregunte al coordinador de una subtransacción hija si puede consumar una subtransacción *subTrans*. El primer argumento *trans* es el identificador de la transacción del nivel superior. El participante responde con su voto *Sí / No*.

Figura 8.10
¿puede Consumar? Para el protocolo de dos fases plano.

puedeConsumar?(trans, listaAbortadas) → *Sí / No*

Llamada desde el coordinador a un participante para preguntar si puede consumar una transacción. El participante responde con su voto *Sí / No*.

Interbloqueos distribuidos

Pueden surgir interbloqueos de un único servidor cuando se utilizan bloqueos para el control de concurrencia. Los servidores deben bien evitarlos, o bien detectarlos y resolverlos. La utilización de tiempos límite para posibles interbloqueos es una aproximación tosca, ya que es difícil elegir un intervalo de tiempo límite apropiado y las transacciones serán abortadas de forma innecesaria.

Con los esquemas de detección de interbloqueos, se aborta una transacción sólo cuando se ha visto involucrada en un interbloqueo. La mayoría de los esquemas de detección de interbloqueos funcionan detectando ciclos en los grafos ***espera por*** de las transacciones.

En teoría, en un sistema distribuido en el que están involucrados múltiples servidores a los que están accediendo múltiples transacciones, puede construirse un grafo *esperar por* global a partir de los grafos *espera por* locales.

Interbloqueos distribuidos

La Figura 8.12 muestra el entrelazado de las transacciones *U, V* y *W* que involucran a los objetos *A* y *B* gestionados por los servidores *X* e *Y,* y los objetos C y *D* gestionados por el servidor Z.

En la Figura 8.13(a) el grafo ***espera por*** completo muestra que un ciclo asociado a un interbloqueo está formado por arcos alternados, que representan a una transacción esperando por un objeto y un objeto retenido por una transacción.

Ya que cualquier transacción puede estar esperando sólo por un objeto cada vez, los objetos pueden dejarse fuera de los grafos *espera por,* como se muestra en la Figura 8.13(b).

Figura 8.12
Entrelazamientos de las transacciones *U*, *V* y *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*

Figura 9.13
Interbloqueo distribuido

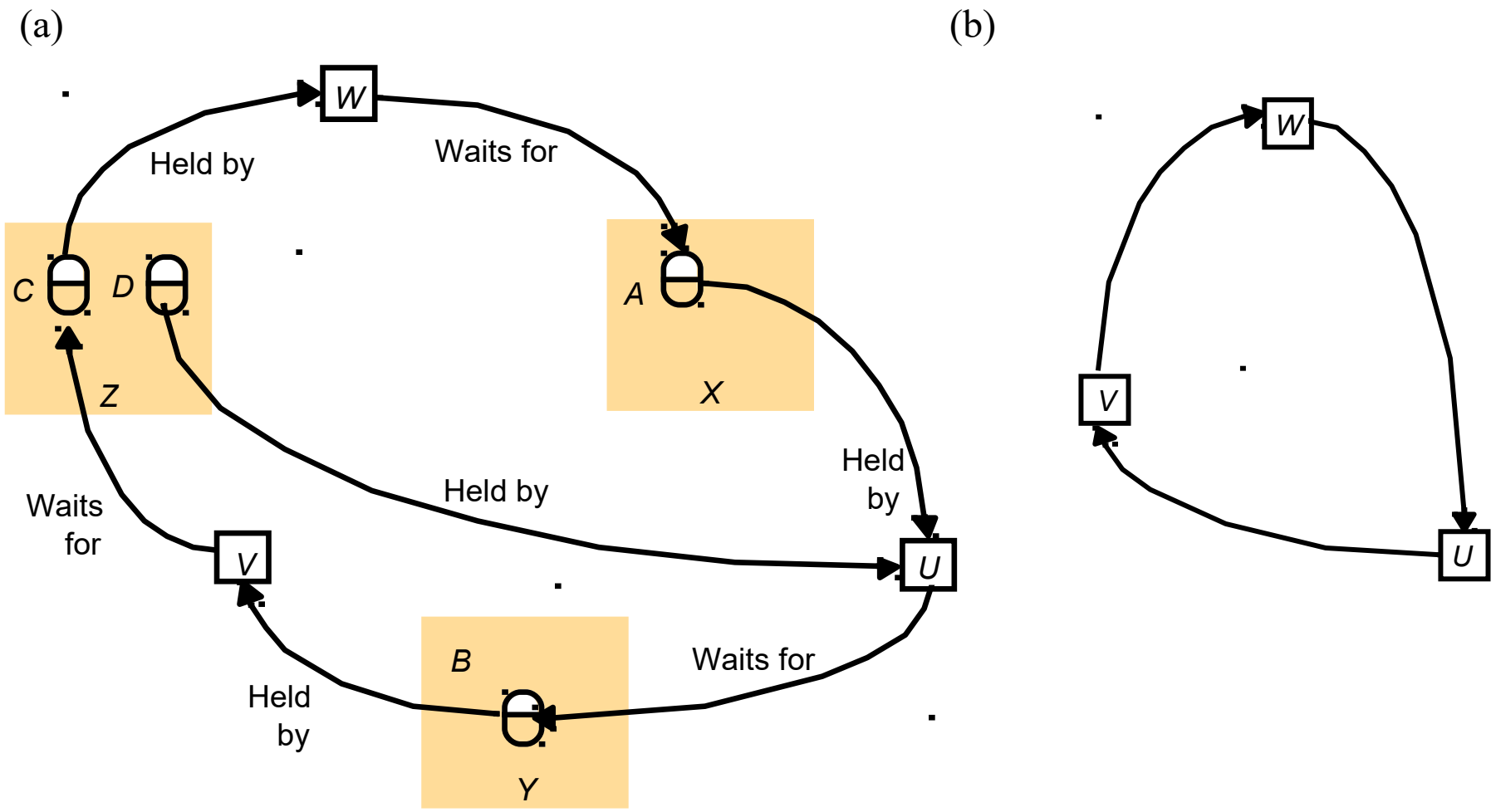
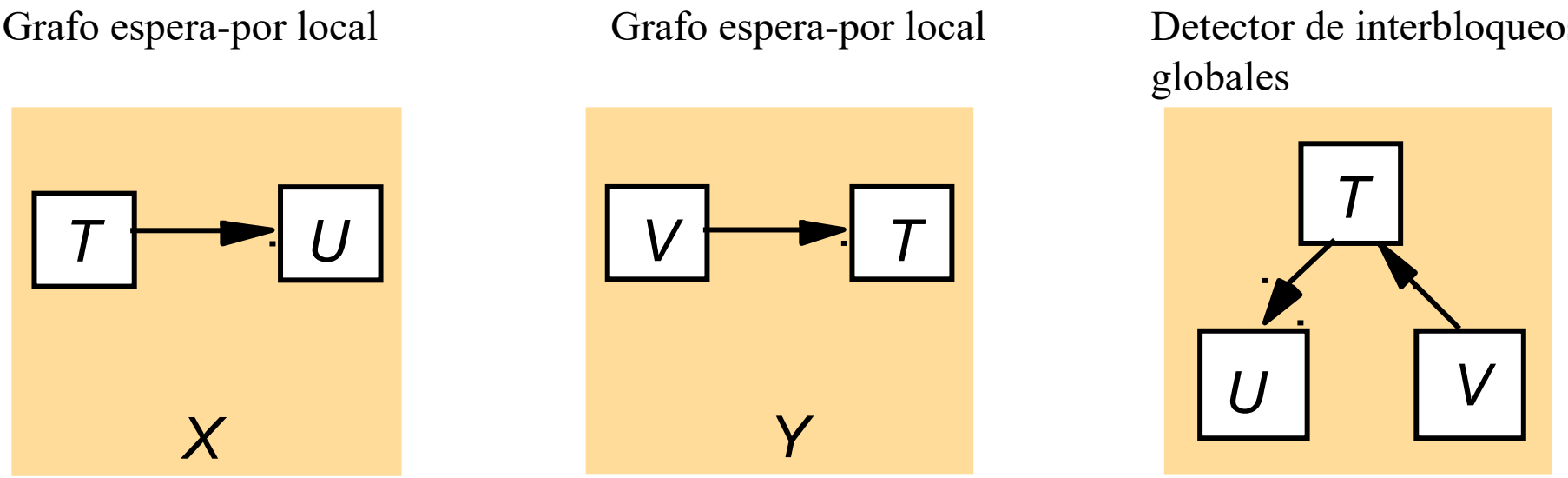


Figura 9.14
Grafos espera por locales y globales



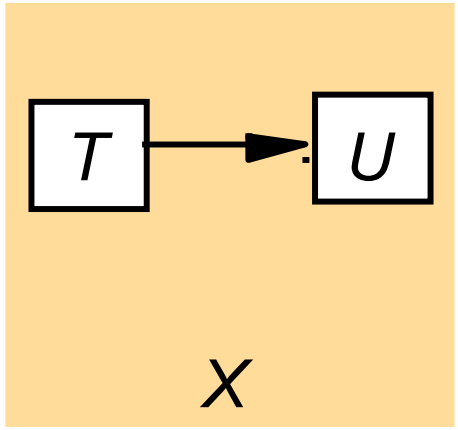
Interbloqueos fantasmas.

Un interbloqueo que se «detecta» pero que realmente no lo es se conoce como interbloqueo fantasma. En la detección de interbloqueos distribuidos, la información acerca de las relaciones ***espera por*** entre las transacciones se transmite de un servidor a otro. Si existe un interbloqueo, la información necesaria se recogerá, al final, en algún lugar y se detectará el ciclo. Ya que este procedimiento tarda un cierto tiempo, existe la posibilidad de tanto, una de las transacciones que mantiene un bloqueo se haya liberado, en cuyo caso ya no existe.

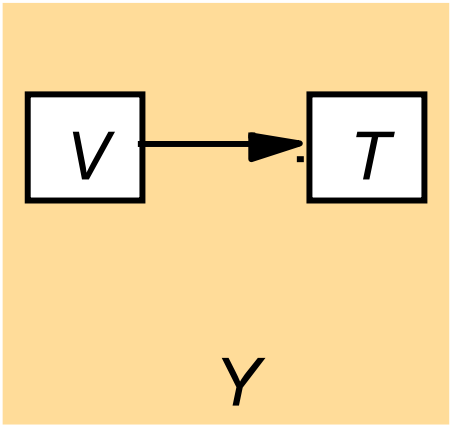
Interbloqueos fantasmas

Considérese el caso de un detector de interbloqueo global, que recibe los grafos *espera por* locales de los servidores *X* e *Y,* como se muestra en la Figura 9.14. Supóngase que la transacción *U* a continuación libera un objeto en el servidor *X* y solicita el que mantiene *V en el servidor Y.* Supóngase además que el detector global recibe el grafo local del servidor Y antes que el del servidor *X.* En este caso, detectaría un ciclo *T- U - V - T,* aunque el arco *T- U* ya no existe. Este es un ejemplo de interbloqueo fantasma.

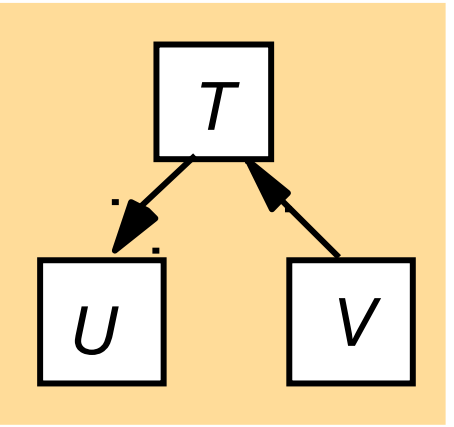
Grafo espera-por local



Grafo espera-por local



Detector de interbloqueos globales



Interbloqueos distribuidos

Caza de arcos. Una aproximación distribuida para la detección de interbloqueos utiliza una técnica denominada captura de los arcos *(edge chasing)* o empuje de caminos *(path pushing).*

En esta aproximación, el grafo no se construye *espera por* global, sino que cada uno de los servidores implicados posee información sobre algunos de sus arcos.

Los servidores intentan encontrar ciclos mediante el envío de mensajes denominados *sondas,* que siguen los arcos de un grafo a través del sistema distribuido.

Un mensaje sonda está formado por las relaciones ***espera por*** de transacciones que representan un camino en el grafo ***espera por*** global.

Interbloqueos distribuidos

Los algoritmos de captura de arcos tienen tres pasos: iniciación, detección y resolución.

Iniciación: cuando un servidor percibe que una transacción T comienza a buscar a otra transacción U, y ésta a su vez está esperando para acceder a un objeto en otro servidor, iniciará la detección enviando una sonda que contiene el arco < T-> U> al servidor del objeto por el cual está bloqueada la transacción U. Si U está compartiendo un bloqueo, se envían sondas a todos los que poseen dicho bloqueo. Algunas veces otras transacciones pueden comenzar a compartir el bloqueo más tarde, en cuyo caso también deben enviárseles sondas a ellas.

Detección: la detección consiste en recibir sondas y decidir si se ha producido un interbloqueo o si hay que enviar nuevas sondas.

Interbloqueos distribuidos

Por ejemplo, cuando un servidor de un objeto recibe la sonda < T-U> (indicando que T está esperando por la transacción U, que retiene un objeto local) comprueba si U está también esperando. Si es así, la transacción por la que espera (por ejemplo, V) se añade a la sonda (consiguiendo < T - U- V>), y si la nueva transacción (V) está esperando por otro objeto en otro sitio, se vuelve a enviar la sonda.

De esta forma, se construye un camino a través del grafo espera-por global añadiendo un eje cada vez. Antes de volver a enviar una sonda, el servidor comprueba si la transacción que acaba de ser añadida (por ejemplo, T) ha ocasionado un ciclo en la sonda (por ejemplo < T - U- V- T>).

Si es éste el caso, ha encontrado un ciclo en el grafo y se ha detectado un interbloqueo.

Interbloqueos distribuidos

Resolución: cuando se detecta un ciclo, se aborta una transacción en el ciclo para romper el interbloqueo.
En nuestro ejemplo, los siguientes pasos describen cómo se inicia la detección de interbloqueos y cómo se envían las sondas durante la correspondiente fase de detección.

. El servidor X inicia la detección enviando la sonda <W-U> al servidor de *B* (Servidor Y).
. El servidor Y recibe la sonda < W- *U* > , observa que *B* es retenida por *V* y concatena V a la sonda para producir < W-U-V> . Observa que *V está* esperando por C en el servidor Z. La sonda se envía al servidor z.
. El servidor Z recibe la sonda < *W* - *U*-*V*> y observa que C es retenida por *W y concatena W a* la sonda para producir < *W* -U- *V*- *W*> .

Este camino contiene un ciclo. El servidor detecta un interbloqueo. Para romper el interbloqueo debe abortarse una de las transacciones en el ciclo.

Figura 8.15
Sondas transmitidas para detectar interbloqueos

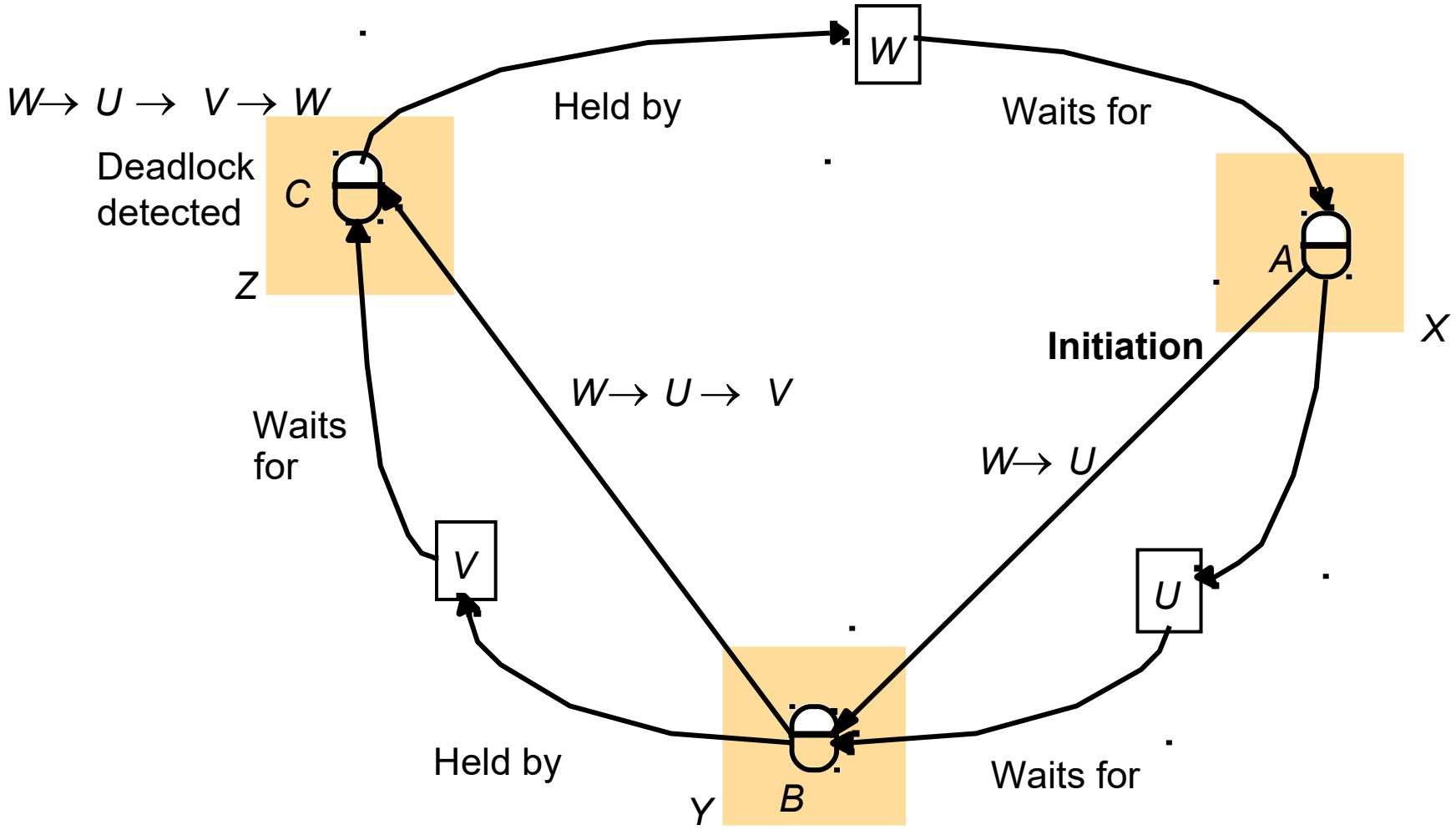
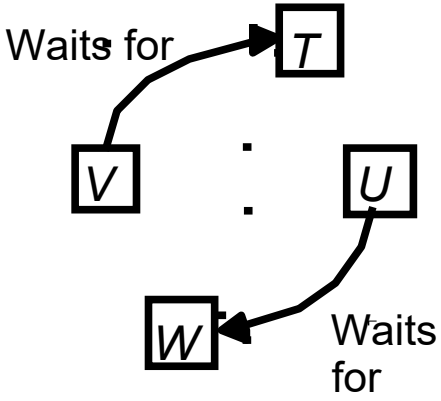
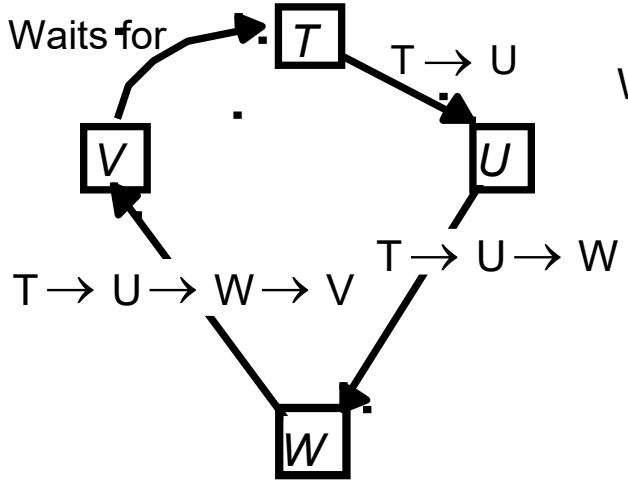


Figura 8.16
Dos sondas iniciadas

(a) Situacion inicial



(b) Deteccion iniciada en un objeto solicitado por T



(c) Deteccion iniciada en un objeto solicitado por W

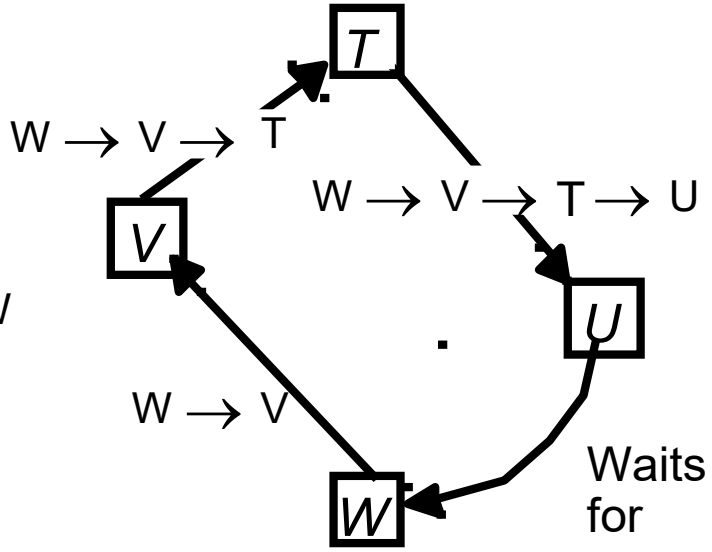
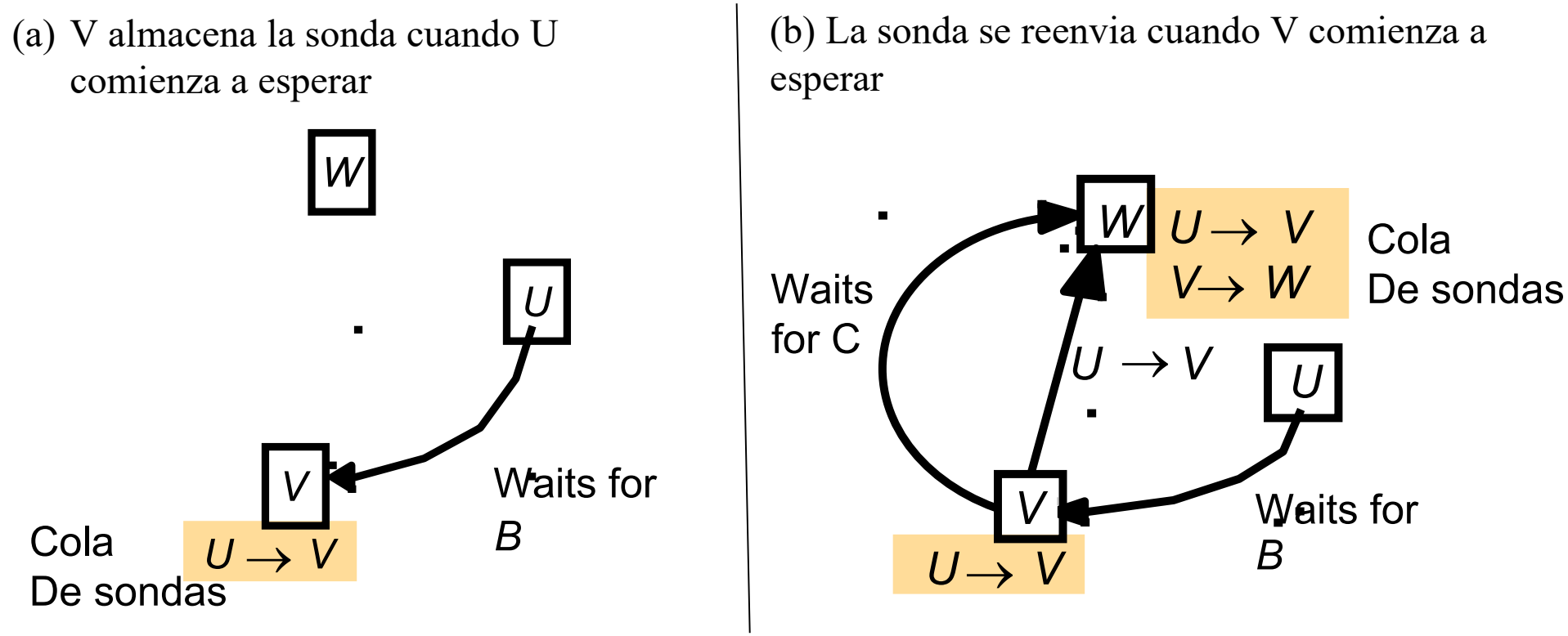


Figura 8.17
Sondas que viajan cuesta abajo



Recuperación de transacciones

Aunque los servidores de archivos y base de datos mantienen los datos en almacenamiento permanente, otros tipos de servidores no necesitan hacerlo, excepto por motivos de recuperación.

Se asume que cuando el servidor está funcionando, mantiene todos sus objetos en memoria volatil y registra sus objetos consumados en un archivo o archivos de recuperación.

Por lo tanto, la recuperación consiste en restaurar el servidor a partir de los dispositivos de almacenamiento permanente y dejarlo con las últimas versiones consumadas de los objetos.

Figura 8.18
Tipos de entrada en un archivo de recuperación

Tipo de entrada	*Descripción de los contenidos de la entrada*
Objeto	Un valor de un objeto
Estado de la transacción	Identificador de la transacción, estado de la transacción (*preparada*, *consumada*, *abortada*), y otros valores de estado utilizados para el protocolo de consumación en dos fases.
Lista de intenciones	Identificador de la transacción y una secuencia de intenciones, cada una de las cuales consiste en <identificador de objeto>, <posición en el archivo de recuperación del valor del objeto>.

Figura 13.18. Tipos de entradas en un archivo de recuperación.

Registro histórico

En la técnica de registro histórico, el archivo de recuperación representa a un registro que contiene el historial de todas las transacciones *realizadas* por el servidor.

El historial consta de *los valores* de *los* objetos, las entradas del estado de la transacción y *las* listas de intenciones de las transacciones.

El orden de las entradas en el registro refleja el orden en el *cual* se prepararon, consumaron o abortaron las transacciones en el servidor. En *la* práctica, el archivo de recuperación contendrá una instantánea reciente de *los valores* de todos *los* objetos en el servidor, seguido de un *historial* de *las* transacciones ocurridas tras dicha instantánea.

Figura 8.19
Registro para un servicio bancario