Sistemas Distribuidos
Módulo 8

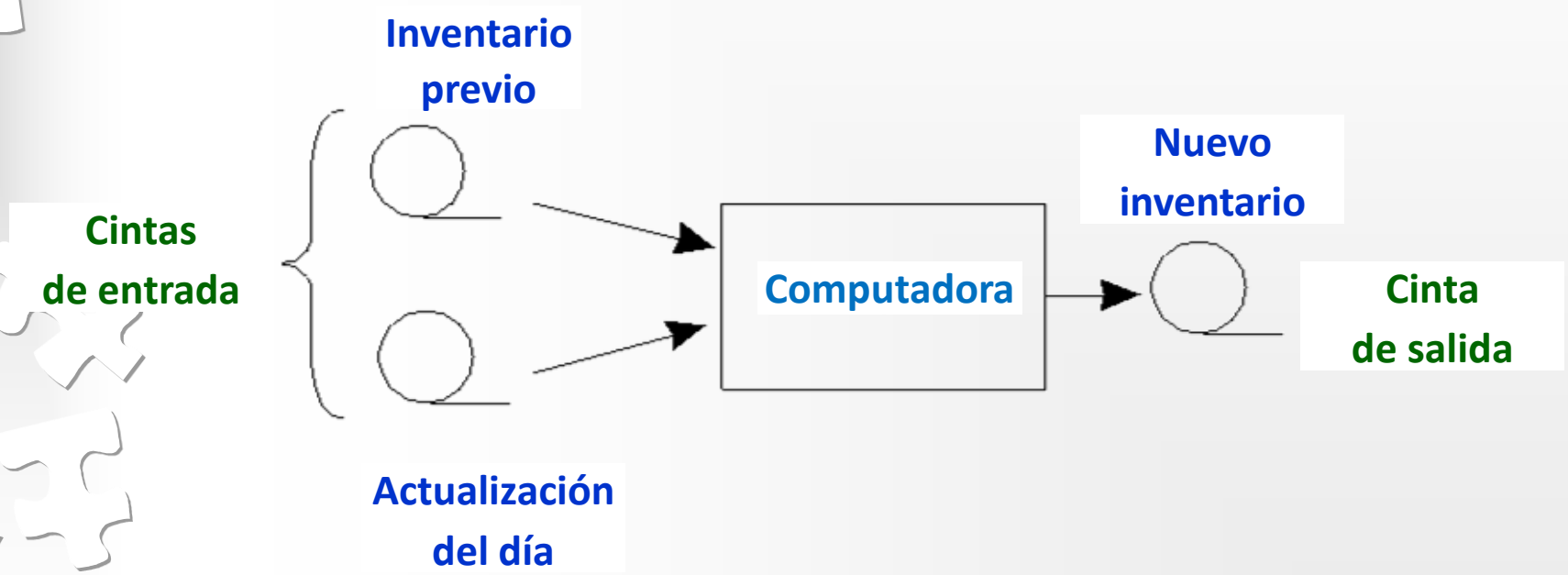
Transacciones Distribuidas

Agenda

1. Modelo
2. Definiciones
3. Propiedades
4. Tipos de Transacciones
    1. Planas
    2. Anidadas
    3. Centralizadas
    4. Distribuidas
5. Protocolo de Consumación de 2 fases anidadas
    1. Jerárquico
    2. Plano

Transacciones Distribuidas - Modelo

La actualización de una cinta maestra es tolerante a las fallas.



Transacciones Distribuidas - Modelo

a) Commit de una Transacción para reservar tres vuelos

b) Aborto de la transacción cuando el tercer vuelo no esta disponible

```
BEGIN_TRANSACTION
  reserve BUE -> JFK;
  reserve JFK -> Nairobi;
  reserve Nairobi -> El Cabo;
END_TRANSACTION
```

(a)

```
BEGIN_TRANSACTION
  reserve BUE -> JFK;
  reserve JFK -> Nairobi;
  reserve Nairobi -> El Cabo full =>
ABORT_TRANSACTION
```

(b)

Transacciones Distribuidas

¿Qué es una transacción?

- Base de datos
- Sistema de archivos
- Objetos
- Cliente

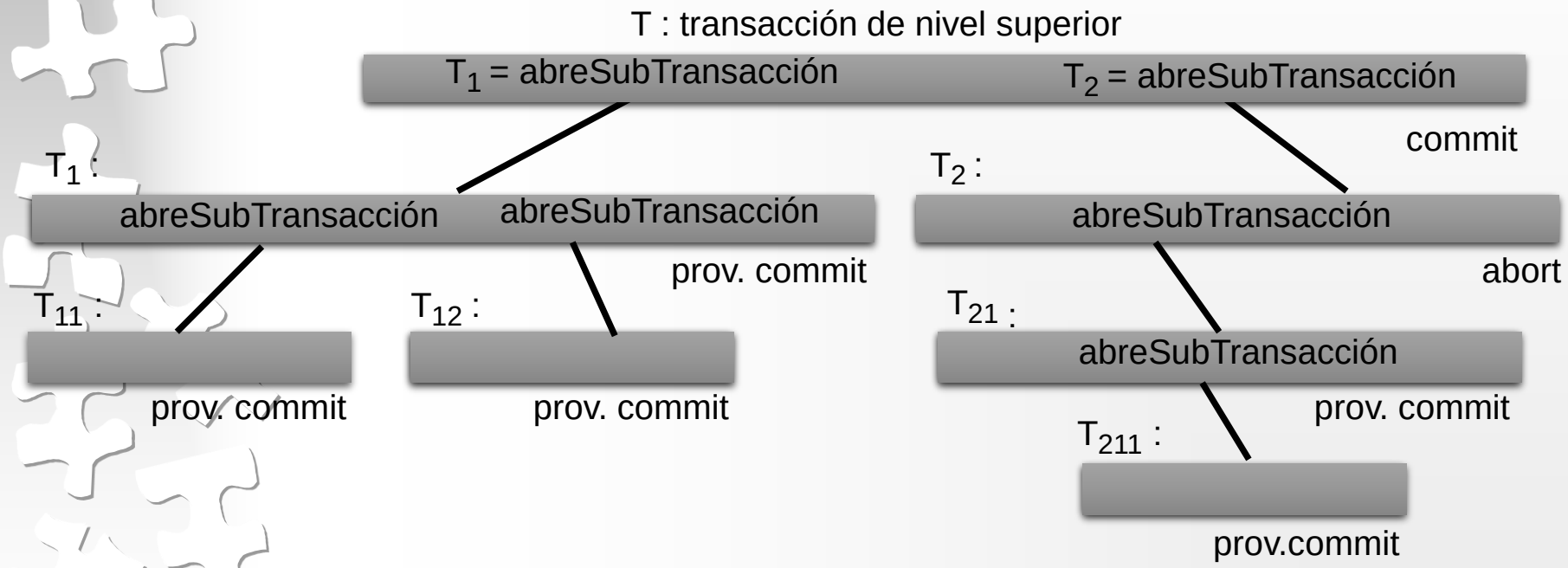
Transacciones Distribuidas

Tipos de Transacciones

- Planas
- Anidadas

- Centralizadas
- Distribuidas

Transacciones Distribuidas - Anidadas



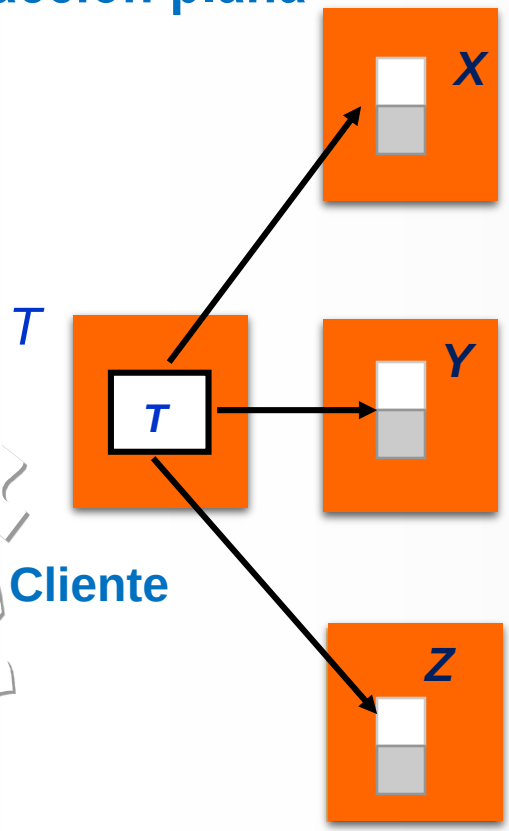
Transacciones Distribuidas

Transacciones distribuidas planas y anidadas

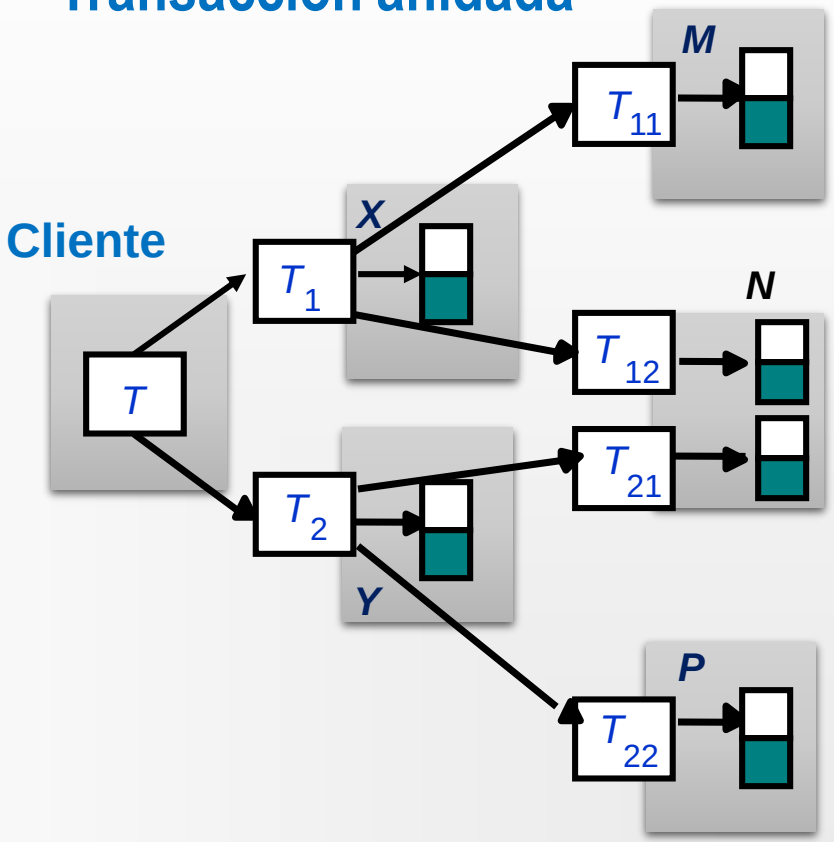
- En una **transacción plana**, el cliente hace requerimientos a más de un servidor. Cada transacción accede a los objetos en los servidores *secuencialmente*.
  - El cliente de la transacción plana espera completar todos sus requerimientos antes de pasar a la próxima.
- En una **transacción anidada**, la transacción de mayor nivel puede abrir subtransacciones y, a su vez cada subtransacción puede abrir otras en niveles más bajos de anidamiento.

Transacciones Distribuidas

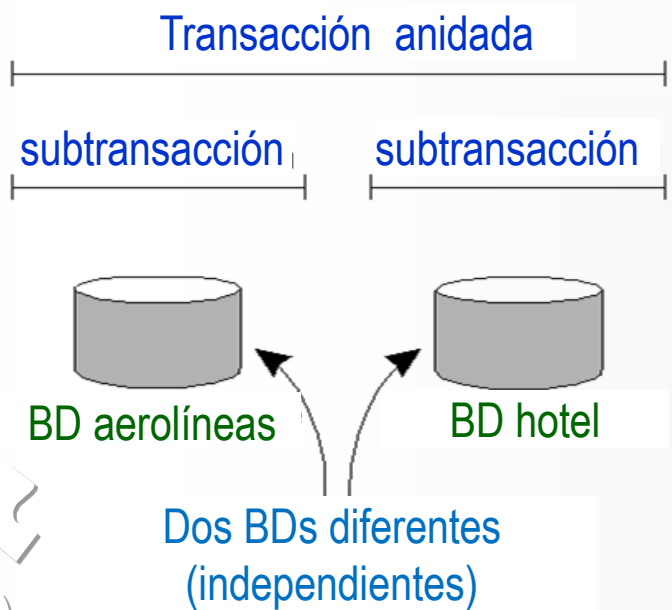
Transacción plana



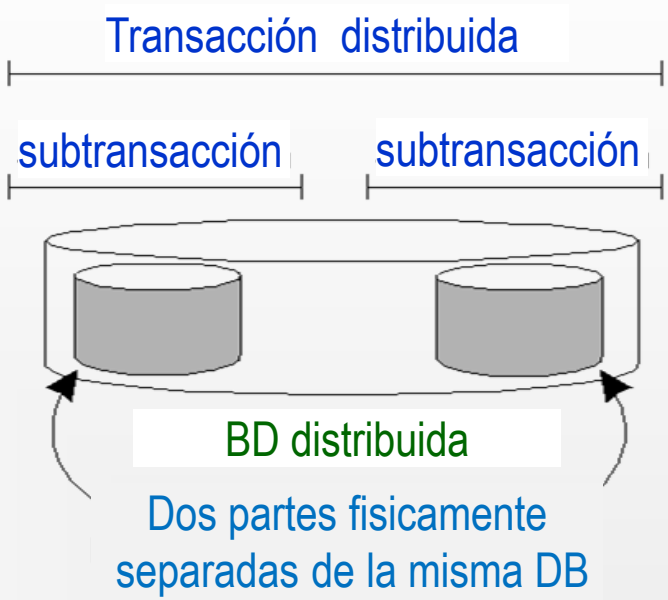
Transacción anidada



Transacciones Distribuidas



Una transacción anidada



Una transacción distribuida

Transacciones Distribuidas - Ejemplo
Transacción bancaria anidada

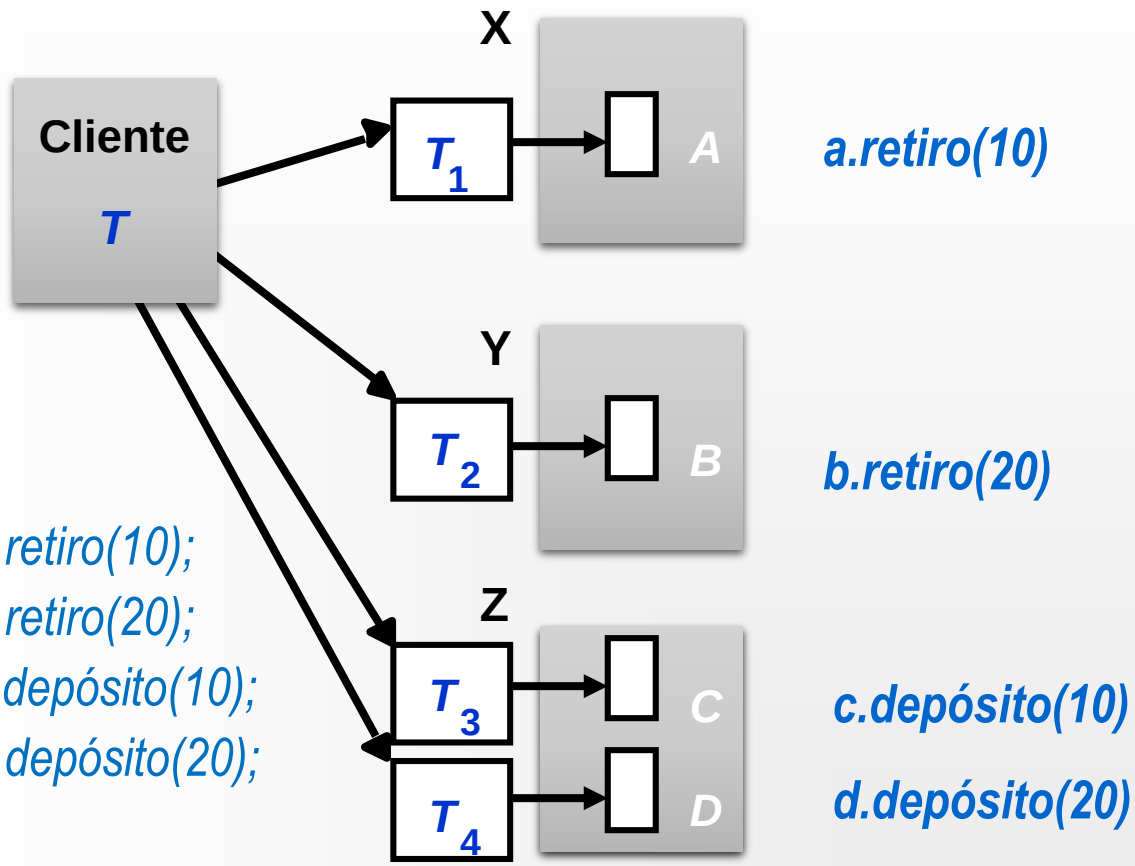
Sea una transacción distribuida donde el cliente transfiere $10 de la cuenta *A* a *C* y $20 de *B* a *D*.

Las cuentas *A* y *B* están separadas en servidores *X* e *Y* y las cuentas *C* y *D* están en el servidor *Z.*

Forma resolución: ***transacción*** se estructura como un conjunto de cuatro ***transacciones anidadas*** (dos depósitos y dos retiros)

Pueden correr en **paralelo** y lograr mayor rendimiento que una transacción plana.

Transacciones Distribuidas



T = abreTransacción
abreSubTransacción a.retiro(10);
abreSubTransacción b.retiro(20);
abreSubTransacción c.depósito(10);
abreSubTransacción d.depósito(20);
cierraTransacción

Transacciones Distribuidas

El coordinador

Los **servidores** que ejecutan requerimientos que son parte de una **transacción distribuida** necesitan **poder comunicarse** con otros para coordinar sus acciones cuando la transacción ***commits***.

- Un cliente comienza la transacción enviando un requerimiento abreTransacción a un coordinador en algún servidor.
- El coordinador que es contactado lleva adelante la abreTransacción y retorna un identificador al cliente (éste debe ser único).
- El coordinador que abre la transacción se convierte en el coordinador para la transacción distribuida.

Transacciones Distribuidas

Participante es cada uno de los servidores que administra un objeto accedido por la transacción.

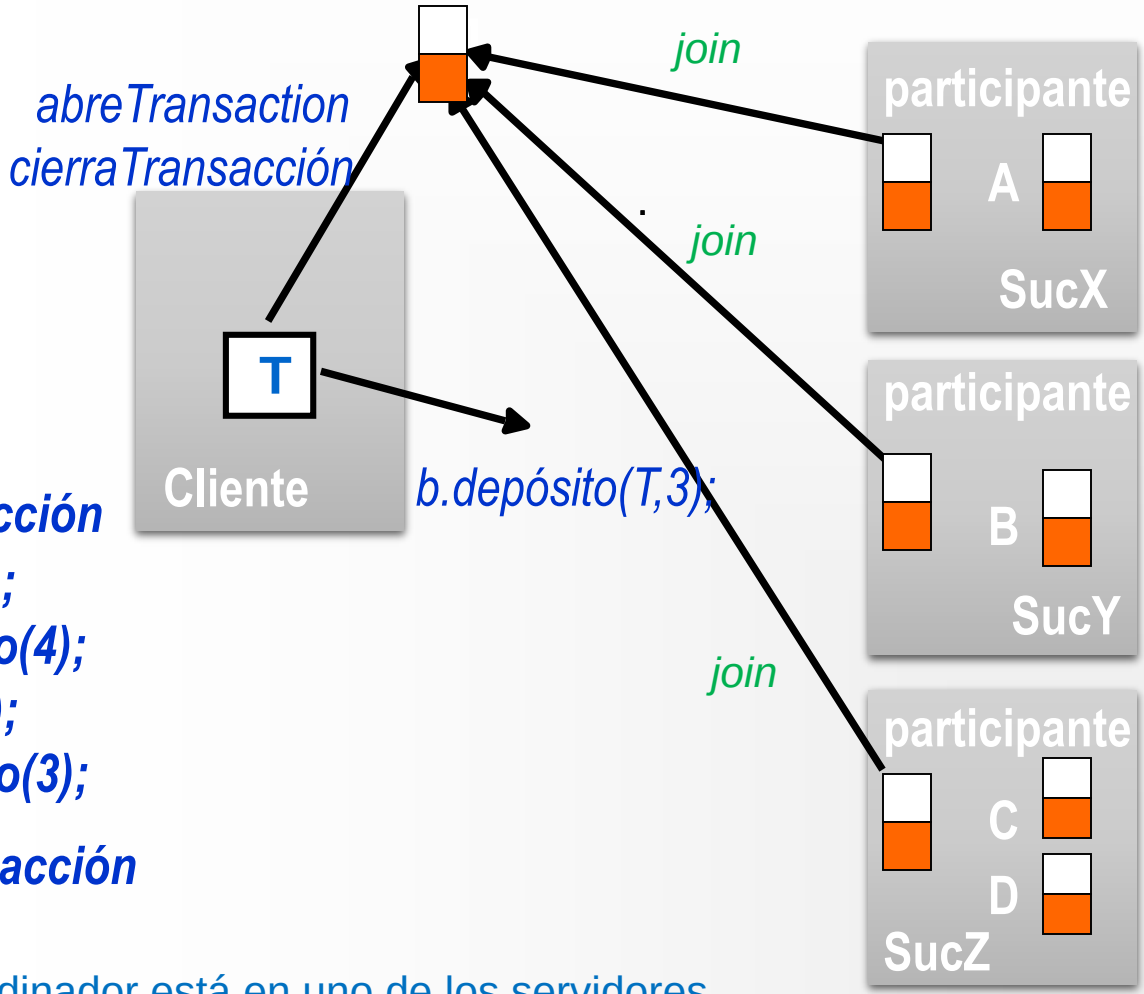
Los participantes son responsables en la cooperación con el coordinador para llevar adelante el protocolo de ***commit*** de la transacción.

Durante el progreso de la transacción, el coordinador registra los participantes y éstos registran al coordinador.

Ejemplo: Un cliente cuya transacción (plana) involucra las cuentas *A, B, C* y D en los servidores *SucX, SucY* y *SucZ*.

La transacción *T* del cliente transfiere $3 de la cuenta *B* a *D* y $4 de *A* a *C*.

Transacciones Distribuidas



T=abreTransacción
a.retiro(4);
c.depósito(4);
b.retiro(3);
d.depósito(3);
cierraTransacción

Nota: el coordinador está en uno de los servidores, p.e. SucX

Transacciones Distribuidas

Protocolos de *Commit* atómicos

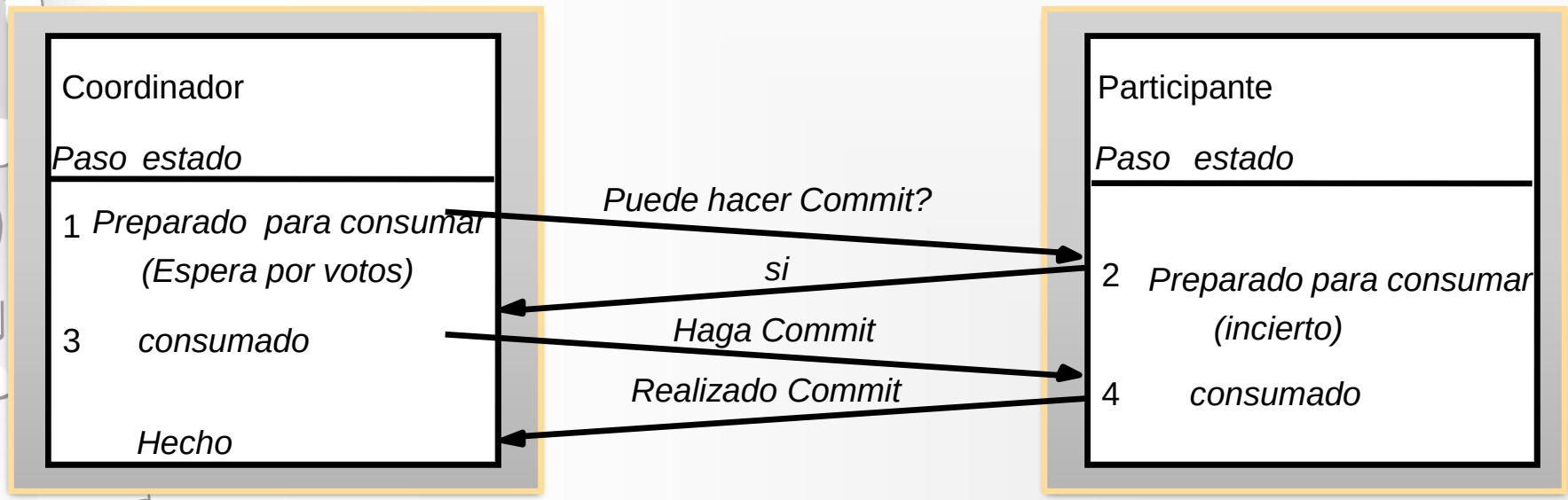
Una transacción llega a su fin cuando los requerimientos del cliente han llegado a *commit* o *abort*.

Los protocolos pueden ser:

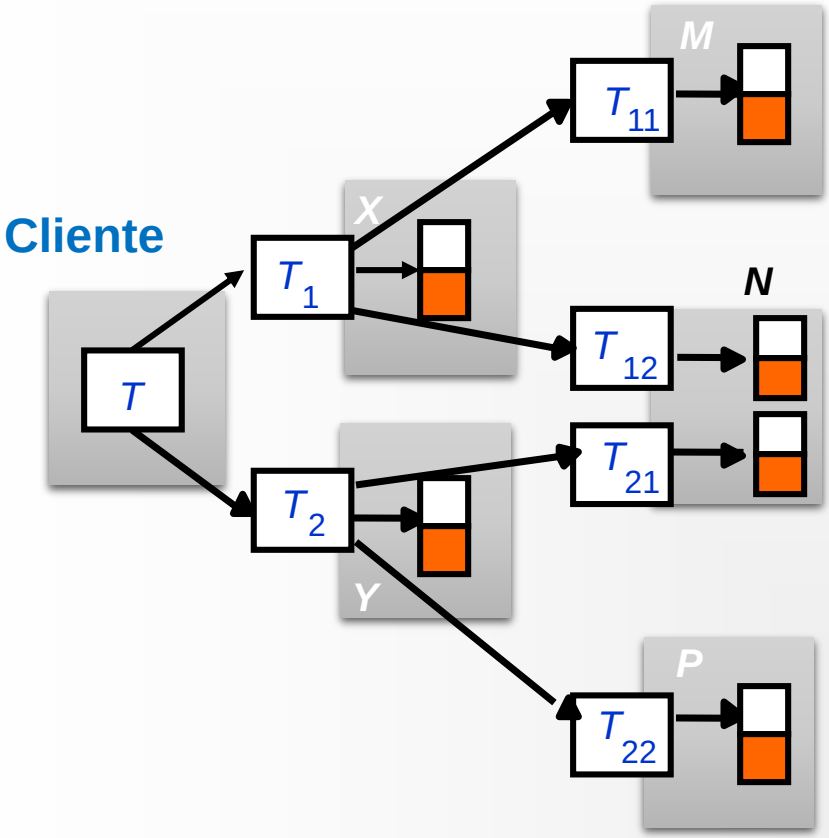
- *Commit* de una fase atómico
- *Commit* de dos fases atómico
- *Commit* de tres fases atómico

Transacciones Distribuidas

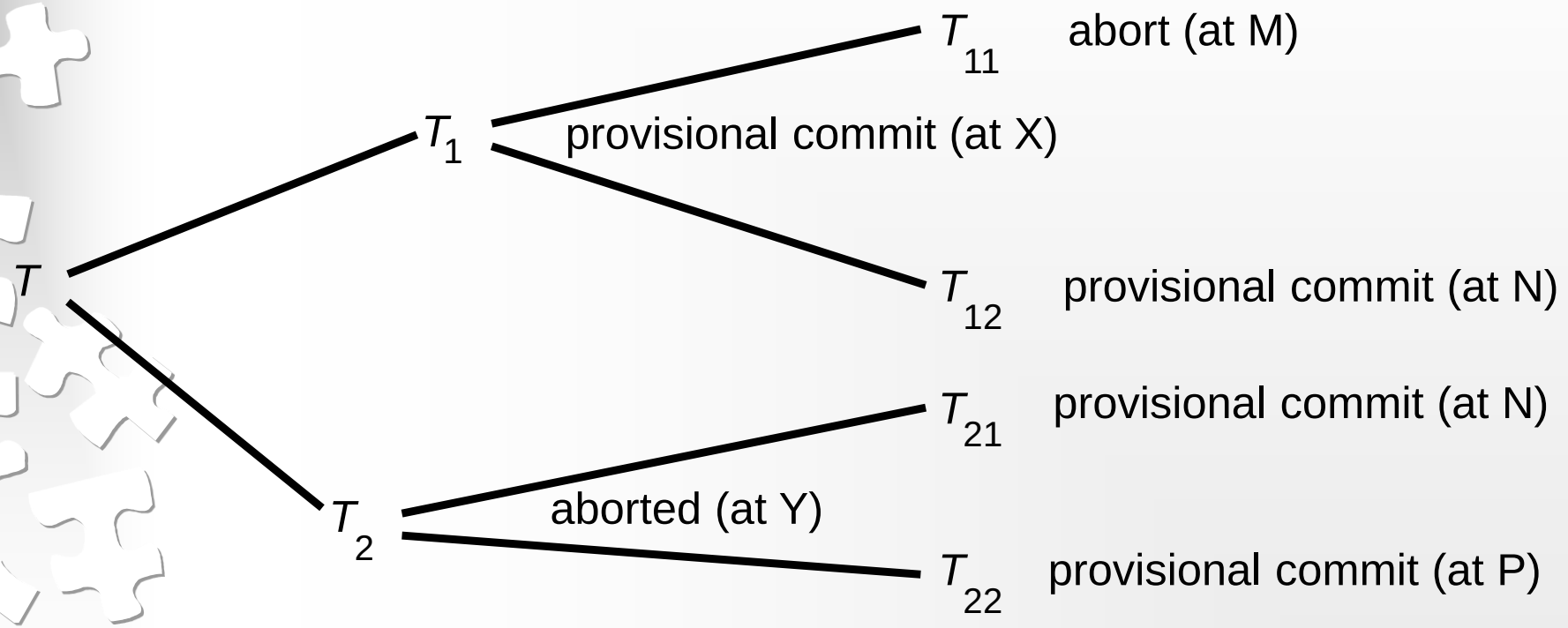
COMUNICACIÓN EN EL PROTOCOLO 2 FASES



Transacciones Distribuidas Anidadas - Ejemplo



Transacciones Distribuidas Anidadas - Commit



Transacciones Distribuidas Anidadas

Coordinador Transacción	*Transacciones Hijas*	*Participantes*	*Lista Prov. Commit*	*Lista Abort*
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 (abortada)		T_2
T_{11}		no (abortada)		T_{11}
T_{12}, T_{21}		T_{12} pero no T_{21}*	T_{21}, T_{12}	
T_{22}		no (padre abortó)	T_{22}	

*T_{21}'s padre ha abortado

Transacciones Distribuidas Anidadas

- Commit para transacciones anidadas

  - Protocolo de consumación en dos fases jerárquico.

    canCommit(trans, subtrans) → si/no

  - Protocolo de consumación en dos fases planos.

    canCommit(trans, abortList) → si/no

Transacciones Distribuidas

- Protocolos de **CONTROL DE CONCURRENCIA** garantizan la consistencia de los datos cuando son accedidos por transacciones concurrentes.
  - Bloqueo
  - Estampillas de Tiempo
  - Concurrencia Optimista

Más información sobre este tema: Sistemas Distribuidos Conceptos y Diseño – 5ta. Edición, Coulouris [Cap. 16]

Transacciones Distribuidas

Recuperación de Transacciones

La propiedad de atomicidad puede describirse en términos de dos aspectos:

- Durabilidad o persistencia.
- Atomicidad ante fallos.

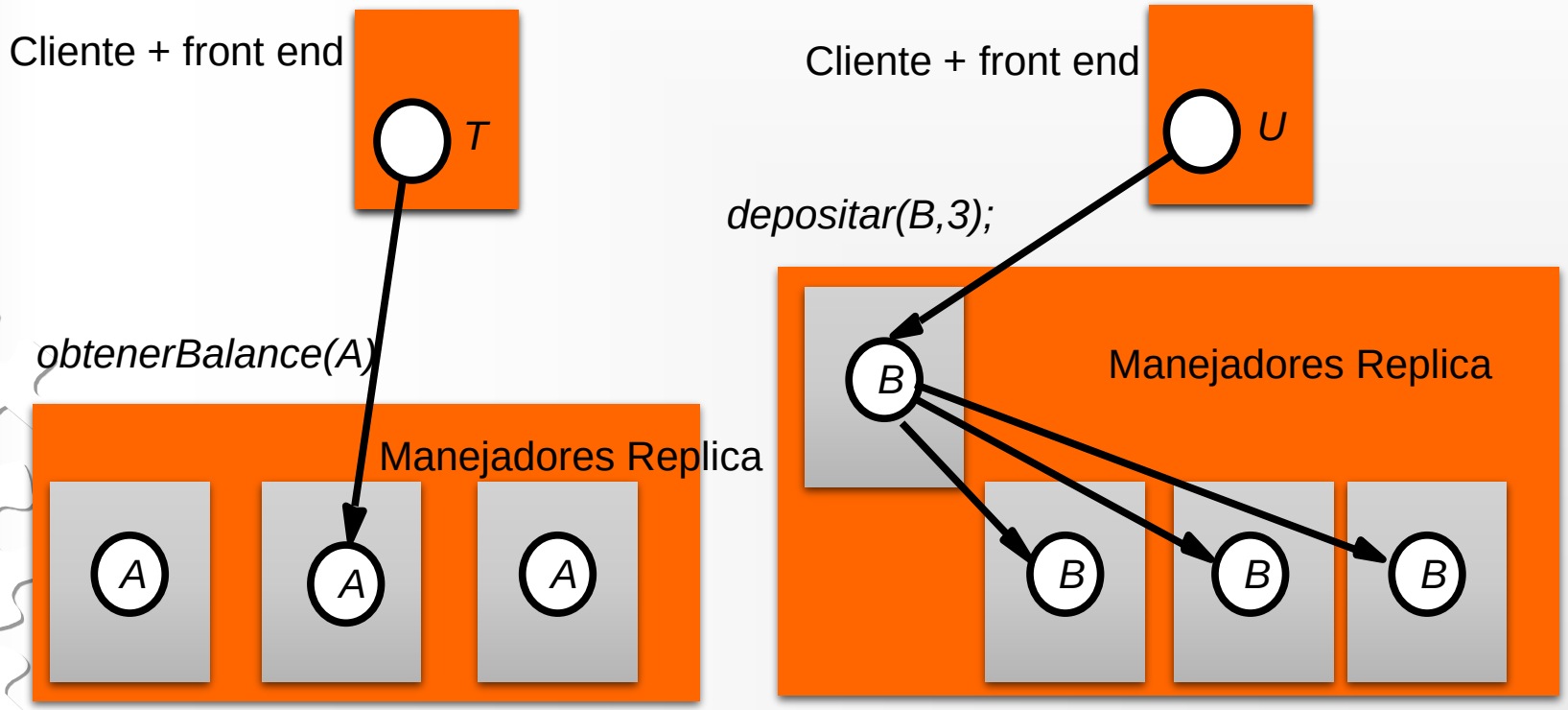
Se asume que cuando un servidor está en funcionamiento, mantiene todos sus objetos en la memoria volátil y registra sus objetos consumados en un ***archivo*** o ***archivo de recuperación***.

Transacciones Distribuidas

Dos aproximaciones al empleo de archivos de recuperación:

- El registro histórico (*logging*).
- Versiones sombra (*shadow*).

Transacciones Distribuidas – Datos Replicados



Bibliografía:

- Tanenbaum, A.S.; van Steen, Maarten; “Distributed Systems: Principles and Paradigms”. 2nd Edition, Prentice Hall, 2007 and 1st Edition 2002.
- Coulouris,G.F.; Dollimore, J. y T. Kindberg; “Distributed Systems: Concepts and Design”. 5th Edition Addison Wesley, 2011.