Kapitel 1

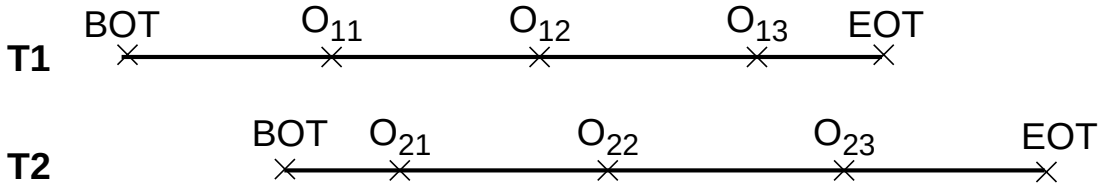
Grundlagen / Wiederholung

Inhalt

- Transaktionen
- Transaktionssysteme
- Zusammenspiel der Transaktionsdienste
- TP-Monitor
- Transaktionen in der Middleware
- Zusammenfassung

Transaktionen (2)

- Ablaufkontrollstruktur: Transaktion



- Eigenschaften
  - Atomicity (Atomarität)
    - TA ist kleinste, nicht mehr weiter zerlegbare Einheit
    - „alles-oder-nichts“-Prinzip
  - Consistency
    - TA hinterläßt einen konsistenten DB-Zustand
    - Zwischenzustände dürfen inkonsistent sein
    - Endzustand muß Integritätsbedingungen erfüllen
  - Isolation
    - Nebenläufige TA dürfen sich nicht gegenseitig beeinflussen
  - Durability (Dauerhaftigkeit)
    - Wirkung erfolgreich abgeschlossener TA bleibt dauerhaft in der DB
    - TA-Verwaltung muß sicherstellen, daß dies auch nach einem Systemfehler gewährleistet ist
    - Wirkung einer erfolgreich abgeschlossenen TA kann nur durch eine sog. kompensierende TA aufgehoben werden

Transaktionen (3)

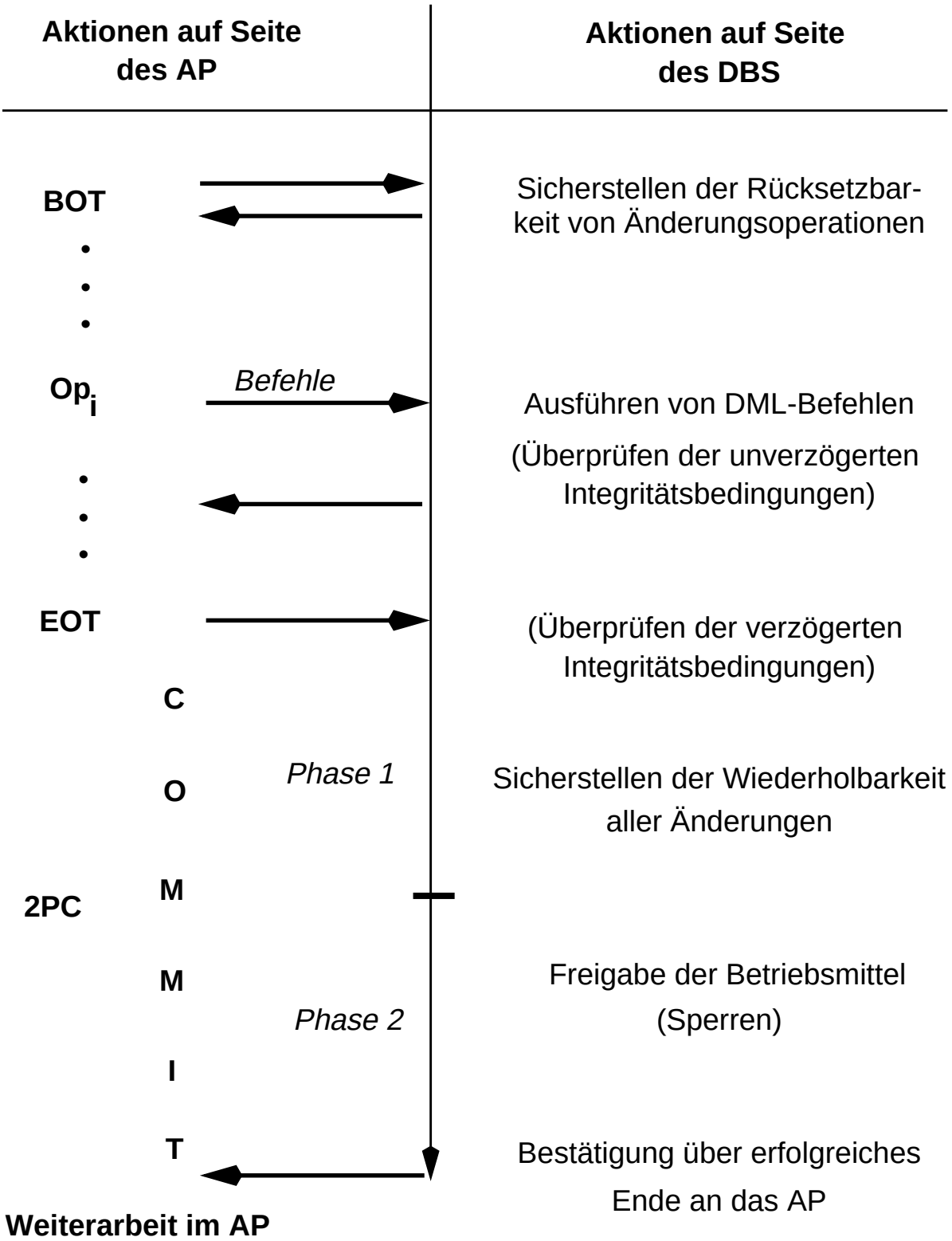
- DB-bezogene Definition der Transaktion:

  Eine TA ist eine ununterbrechbare Folge von DML-Befehlen, welche die Datenbank von einem logisch konsistenten Zustand in einen neuen logisch konsistenten Zustand überführt.

- Transaktionsverwaltung
  - koordiniert alle DBS-seitigen Maßnahmen, um ACID zu garantieren
  - besitzt zwei wesentliche Komponenten
    - Synchronisation
    - Logging und Recovery
  - kann zentralisiert oder verteilt (z.B. bei VDBS) realisiert sein
  - soll Transaktionsschutz für heterogene Komponenten bieten

Transaktionen (4)

❏ Schnittstelle zwischen AP und DBS



Transaktionen (5)

- Integritätskontrolle
  - modellinherente Integritätsbedingungen (relationale Invarianten)
  - semantische Integritätsbedingungen
  - Trigger
  - Regeln
- semantische Integritätsbedingungen
  - Reichweite
    - Attribut
    - Tupel
    - Relation
    - mehrere Relationen
  - Zeitpunkt der Überprüfung
    - immediate
    - deferred
  - Art der Überprüfbarkeit
    - Zustand
    - Übergang
  - Anlaß für Überprüfung
    - Datenänderung
    - Zeitpunkt

Transaktionen (6)

- Synchronisation
  - Ziel: Vermeidung von Anomalien
    - Abhängigkeit von nicht freigegebenen Änderungen *(dirty read)*
    - Verlorengegangene Änderung *(lost update)*
    - Inkonsistente Analyse *(non-repeatable read)*
    - Phantom-Problem
  - Formales Korrektheitskriterium: *Serialisierbarkeit:*

    *Die parallele Ausführung einer Menge von TA ist **serialisierbar**, wenn es eine serielle Ausführung derselben TA-Menge gibt, die den **gleichen DB-Zustand** und die **gleichen Ausgabewerte** wie die ursprüngliche Ausführung erzielt.*
  - Verfahren
    - Sperrverfahren
    - optimistische Verfahren
    - Zeitmarkenverfahren
    - Mehrversionenverfahren
    - Spezialverfahren

Transaktionen (7)

❑ Synchronisation (Forts.)

➪ Verfahren (Forts.)

- Beispiel: Sperrverfahren

  - 2PL-Regeln:

    1. vor jedem Objektzugriff muß Sperre mit ausreichendem Modus angefordert werden
    2. gesetzte Sperren anderer TA sind zu beachten
    3. eine TA darf nicht mehrere Sperren für ein Objekt anfordern
    4. ***Zweiphasigkeit:*** Wachstumsphase vor Schrumpfungsphase (Beachte Varianten: *strikt* und *preclaiming*)
    5. spätestens bei Commit sind alle Sperren freizugeben

  - Beispiel: Hierarchische Sperren

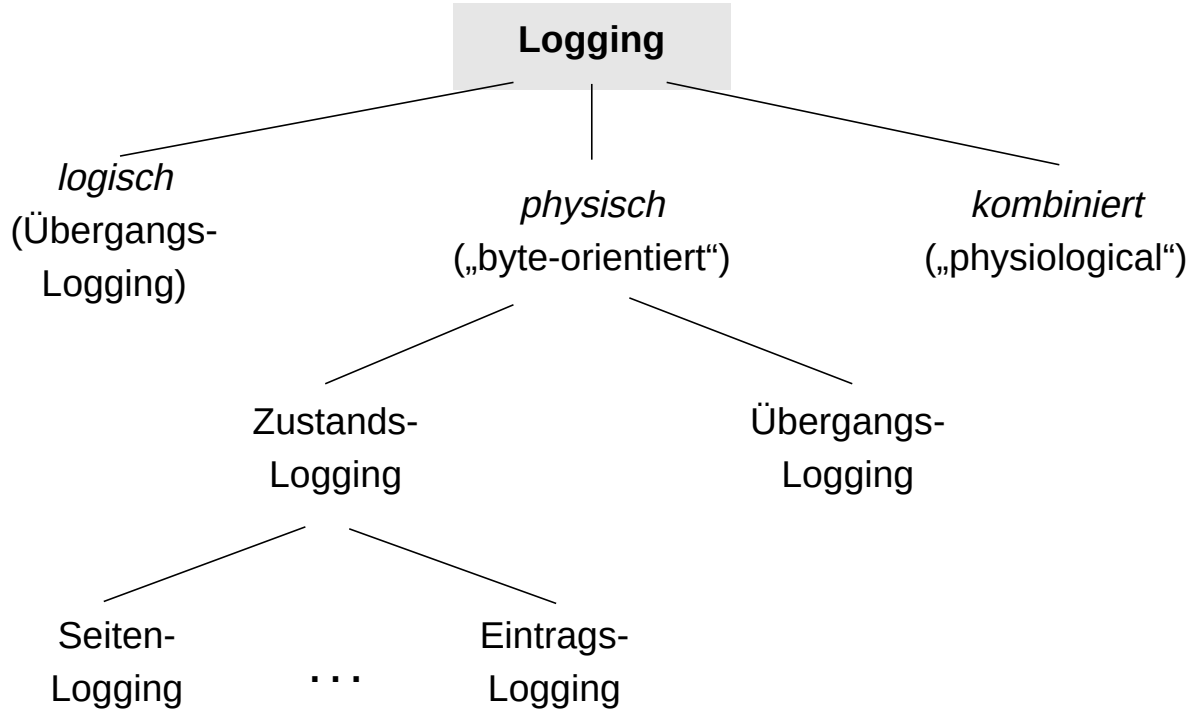
	IR	IX	R	RIX	U	X
IR	+	+	+	+	-	-
IX	+	+	-	-	-	-
R	+	-	+	-	-	-
RIX	+	-	-	-	-	-
U	-	-	+	-	-	-
X	-	-	-	-	-	-

Transaktionen (8)

- Logging und Recovery
  - Recovery-Arten
    - Transaktions-Recovery
      - vollständiges Zurücksetzen (TA-UNDO)
      - partielles Zurücksetzen auf Rücksetzpunkt (Savepoint)
    - Crash-Recovery nach Systemfehler
      - (partielles) REDO für erfolgreiche Transaktionen (Wiederholung verlorengegangener Änderungen)
      - UNDO aller durch Ausfall unterbrochenen Transaktionen (Entfernen der Änderungen aus der permanenten DB)
    - Medien-Recovery nach Gerätefehler
      - Spiegelplatten
      - vollständiges Wiederholen (REDO) aller Änderungen auf einer Archivkopie
    - Katastrophen-Recovery
      - Nutzung einer aktuellen DB-Kopie in einem “entfernten” System oder
      - stark verzögerte Fortsetzung der DB-Verarbeitung mit repariertem/neuem System auf der Basis gesicherter Archivkopien (Datenverlust!)

Transaktionen (9)

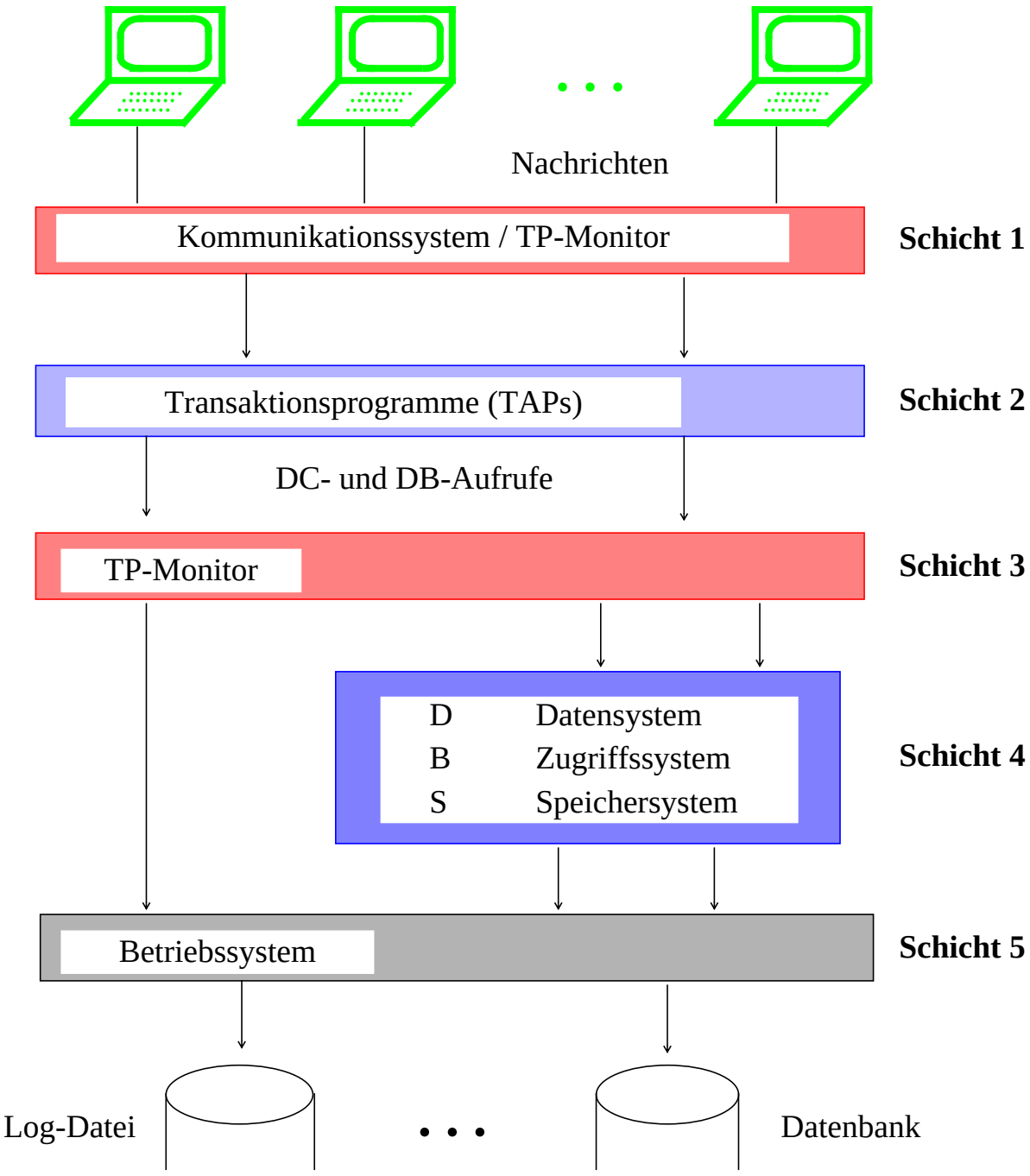
- Logging und Recovery (Forts.)
  - Logging-Verfahren



  - Abhängigkeiten zu anderen Systemkomponenten
    - Einbringstrategie
      - direkt (NON-ATOMIC, Update-in-Place)
      - verzögert (ATOMIC, Bsp.: Schattenspeicherkonzept)
    - DB-Pufferverwaltung
      - Verdrängen ‚schmutziger‘ Seiten (STEAL vs. NOSTEAL)
      - Ausschreibstrategie für geänderte Seiten (FORCE vs. NOFORCE)
    - Empfehlenswert: NON-ATOMIC / STEAL / NOFORCE
    - Sperrverwaltung
      - Log-Granulat kleiner gleich Sperrgranulat!

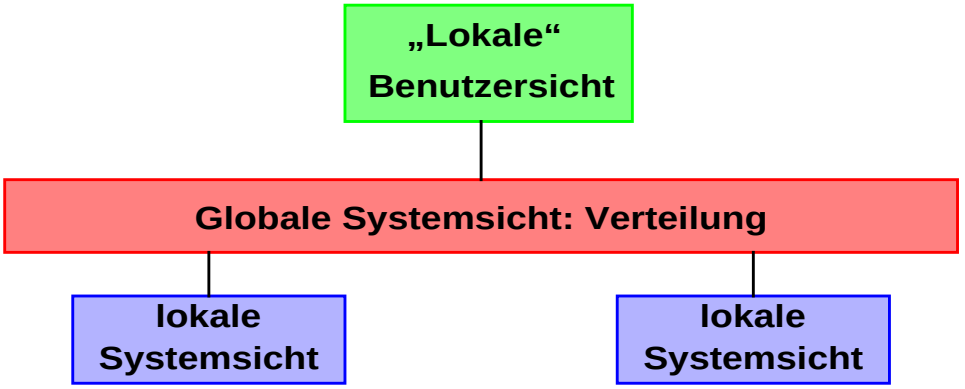
Transaktionssysteme (1)

❏ Schichtenmodell eines Transaktionssystems



Transaktionssysteme (2)

- Wichtige Aspekte eines (verteilten) **TP-Systems**
  - verteilt, heterogen
  - Werkzeuge zur Unterstützung der Anwendungsprogrammierung, -ausführung und -verwaltung
  - Anfragen und Änderungsanforderungen werden über ein Netz von Geräten weitergeleitet und mit Hilfe von DBS verarbeitet
  - Ausgaben treiben Aktuatoren und Geräte, die den Zustand der Miniwelt verändern oder kontrollieren
  - Anwendungen, Datenbanken und Netzwerke wachsen oft über mehrere Jahrzehnte
  - Es wird ein unterbrechungsfreier Betrieb verlangt (continuous operation, keine geplanten Abschaltzeiten)
  - strikte Antwortzeitanforderungen
  - **TP-Monitor** verkörpert Menge zentraler Dienste, die den Fluß der TA durchs System verwalten und koordinieren
  - Prinzipien: Funktions-, Daten- und Lastverteilung
  - gewünschte Sichtbarkeit



Transaktionssysteme (3)

- Klassifikation
  - Verteilung unter Kontrolle des TP-Monitors *(Verteilte DC-Systeme)*
    - TP-Monitor verbirgt weitgehend Heterogenität bezüglich Kommunikationsprotokollen, Netzwerken, BS und Hardware
    - DBS bleiben weitgehend unabhängig (keine Kooperation zwischen DBS)
    - heterogene DBS möglich: Einsatz von DBS-Gateways
    - geringe Implementierungskomplexität
  - Drei wesentliche Alternativen:

1. Transaction Routing

- globale Sicht in Schicht 1 (Kommunikationssystem)
- Einheit der Verteilung ist die Transaktion (TA-Typ)

1. Programmierte Verteilung

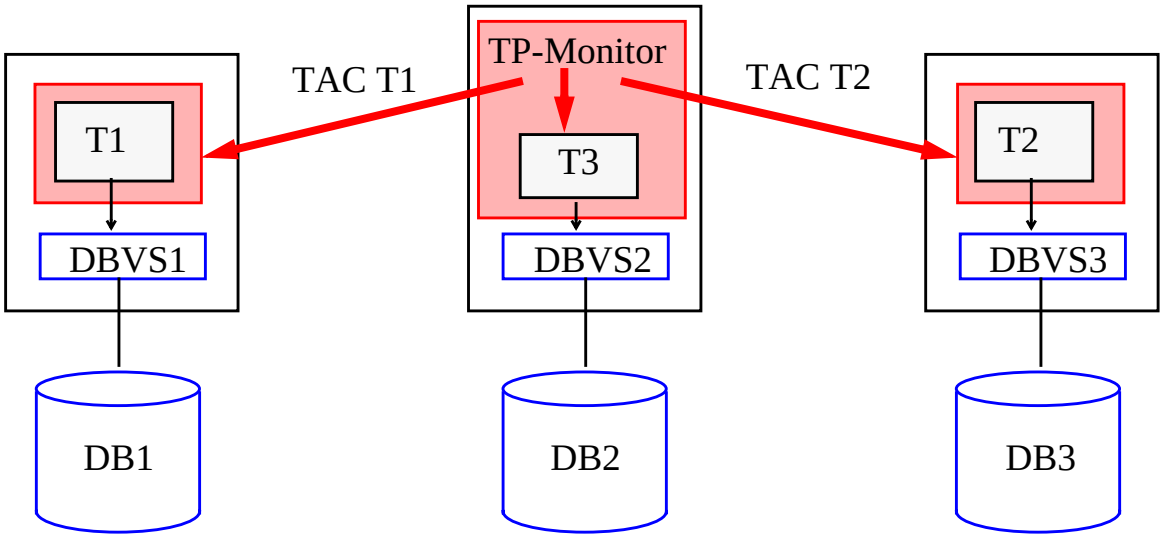
- globale Sicht in Schicht 2 (im AP)
- Einheit der Verteilung ist eine Teil-Transaktion (Programmfragment)

1. Verteilung von DB-Operationen

- globale Sicht in Schicht 3 (TP-Monitor)
- Einheit der Verteilung ist ein DML-Befehl

Transaktionssysteme (4)

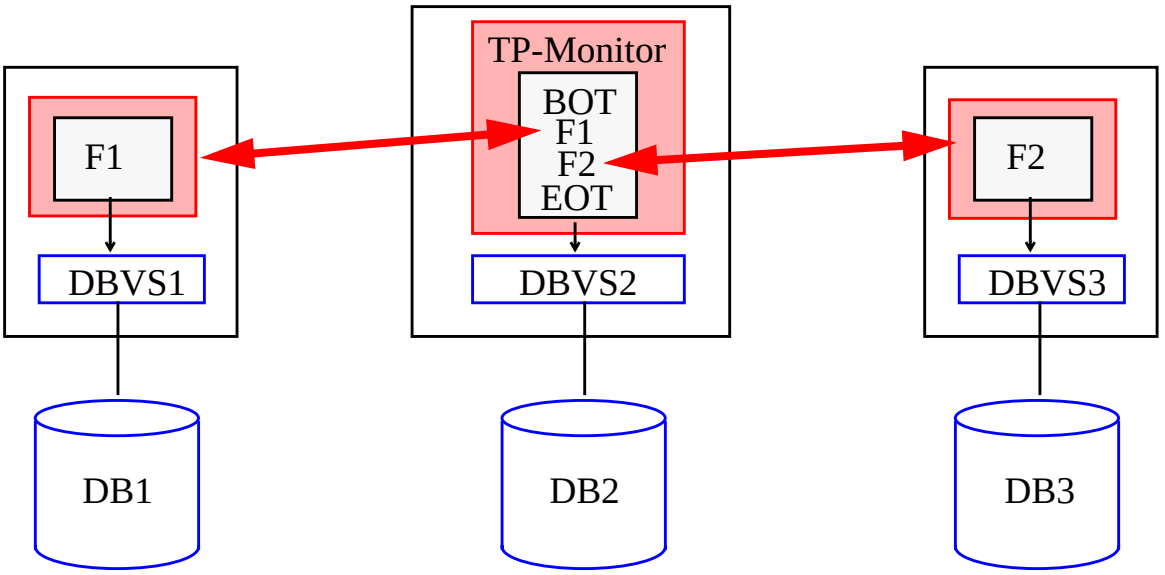
- Transaction Routing
  - Prinzip
    - jeder TA-Typ ist einem Rechner fest zugeordnet
    - TP-Monitor kennt TA-Typ-Zuordnung: Erkennung und Weiterleitung des TAC (→ Ortstransparenz)
    - lokale Transaktionsausführung innerhalb eines Rechners
    - Ausgabenachricht muß ggf. über Zwischenrechner an Benutzer zurückgesendet werden



  - Keine Kooperation zwischen DBVS
    - pro Rechner eigene DB/Schemata
    - heterogene DBS möglich
  - Als alleiniges Verteilgranulat zu inflexibel (keine echt verteilte Transaktionsausführung)
  - Beispiel: IMS MSC (*multiple systems coupling*)

Transaktionssysteme (5)

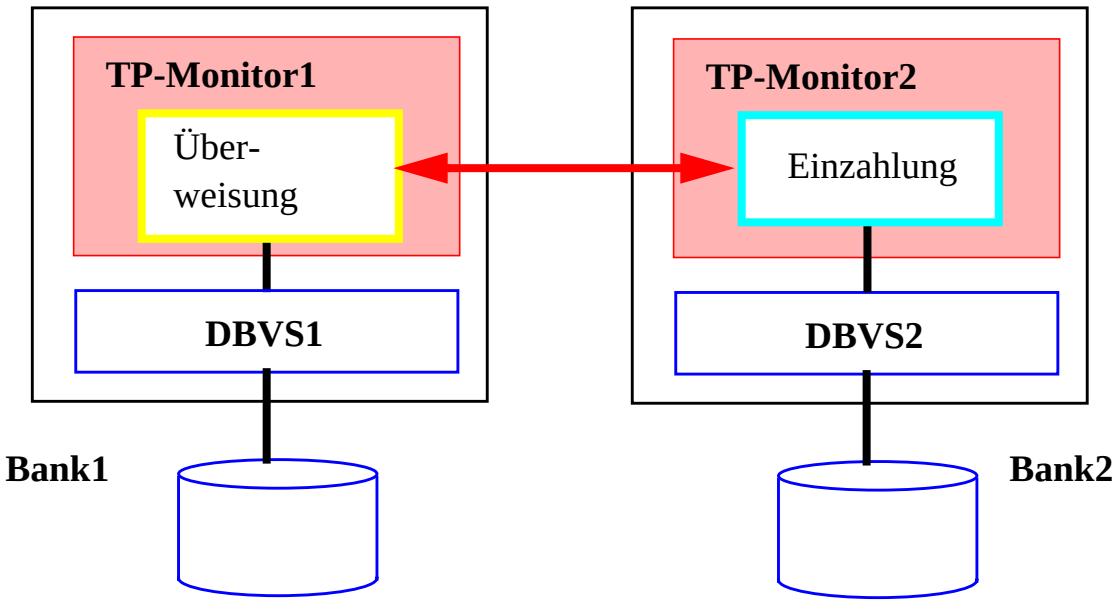
- Programmierte Verteilung
  - Verteilung auf Ebene von Anwendungsprogrammen
    - Zugriff auf externe DB durch Aufruf eines Teilprogramms (RPC) auf dem betreffenden Rechner
    - jedes Teilprogramm greift nur auf lokale DB zu (mit jeweiliger Anfragesprache)



  - Transaktionsverwaltung
    - TP-Monitor koordiniert verteiltes Commit-Protokoll
    - DBS müssen DB-Operationen von nicht-lokalen Transaktionen akzeptieren sowie am Commit-Protokoll teilnehmen
    - Auflösung globaler Deadlocks über Timeout
  - Ortstransparenz kann prinzipiell durch TP-Monitor erreicht werden
  - *Beispiele:* Tandem Pathway, UTM-D, CICS Distributed Transaction Processing

Transaktionssysteme (6)

- Programmierte Verteilung (Forts.)
  - Beispiel: Überweisung zwischen unabhängigen Banken



Eingabenachricht (Parameter) lesen

BEGIN WORK

{ Zugriff auf lokale DB zur Abhebung des Betrags }

```
UPDATE ACCOUNT
SET BALANCE = BALANCE - :S
WHERE ACCT_NO = :K1;
• • •
```

CALL EINZAHLUNG (Bank2, S, K2)

COMMIT WORK

Ausgabenachricht ausgeben

Parameter übernehmen

• • •

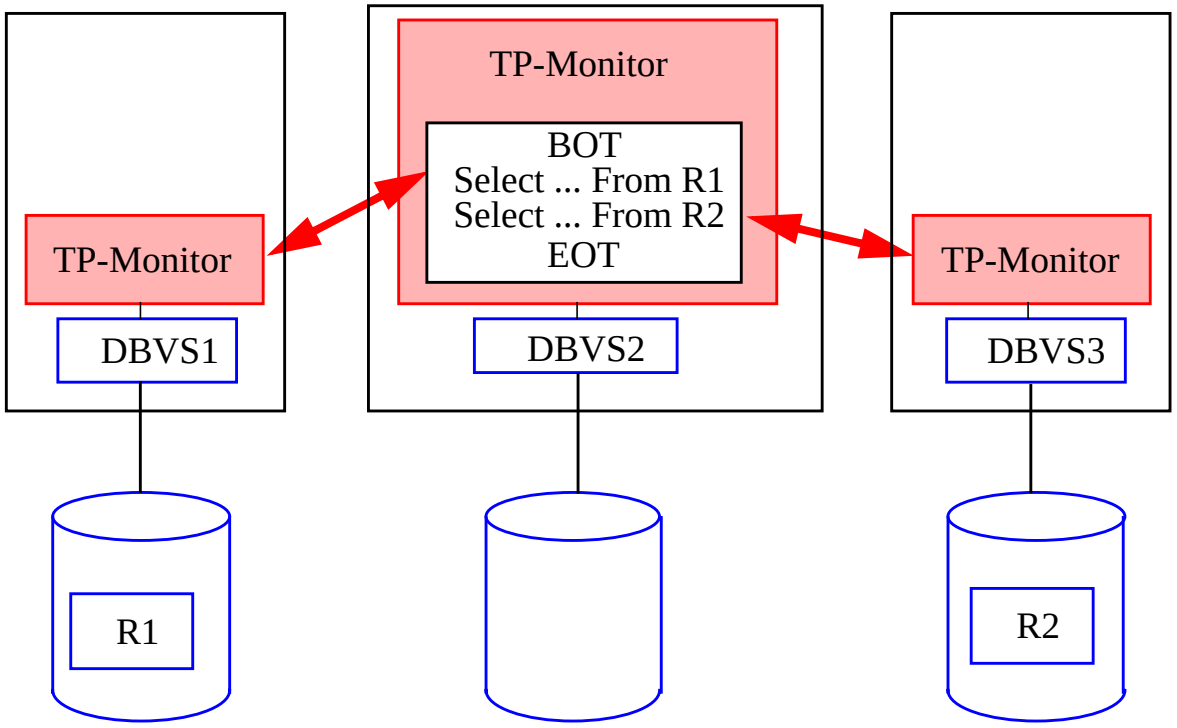
```
UPDATE GIROKTO
SET KSTAND := KSTAND+ :S
WHERE KNUMMER = :K2;
```

• • •

Ausführung zurückmelden

Transaktionssysteme (7)

❏ Verteilung von DB-Operationen (durch TP-Monitor)



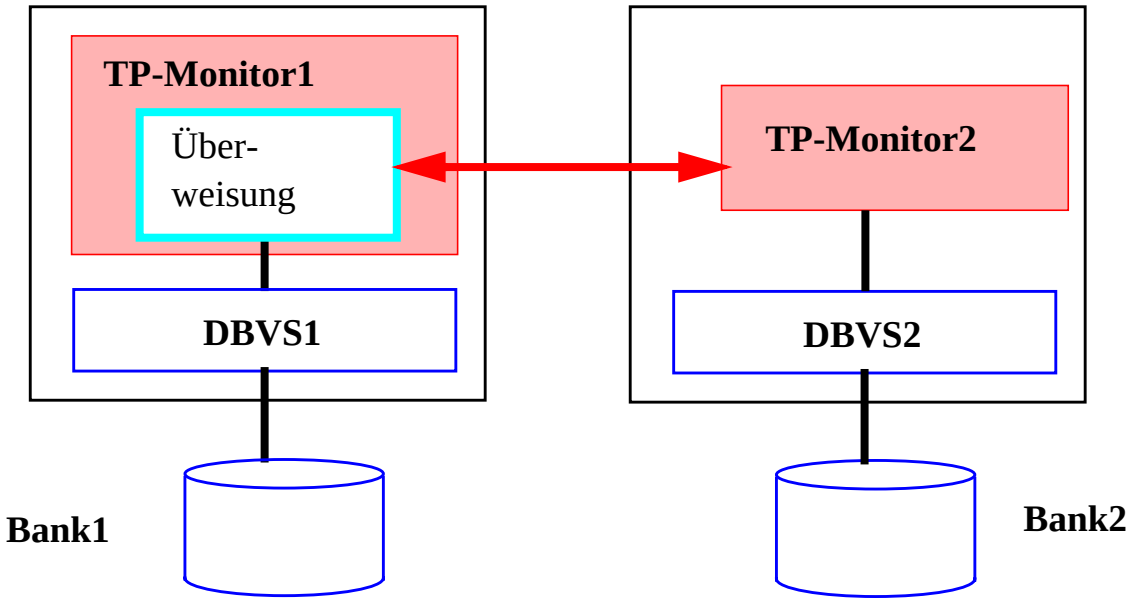
➪ DB-Operationen als Verteileinheiten

- AP können auf entfernte Datenbanken zugreifen; Aufrufweiterleitung durch *Function Request Shipping*
- Programmierer sieht mehrere Schemata; jede Operation muß innerhalb eines Schemas (DB-Partition) ausführbar sein
- gemeinsame Anfragesprache wünschenswert
- Ort der Verteilung kann für AP prinzipiell transparent gehalten werden
- Transaktionsverwaltung im Prinzip wie bei Programmierter Verteilung

➪ Beispiel: CICS *Function Request Shipping*

Transaktionssysteme (8)

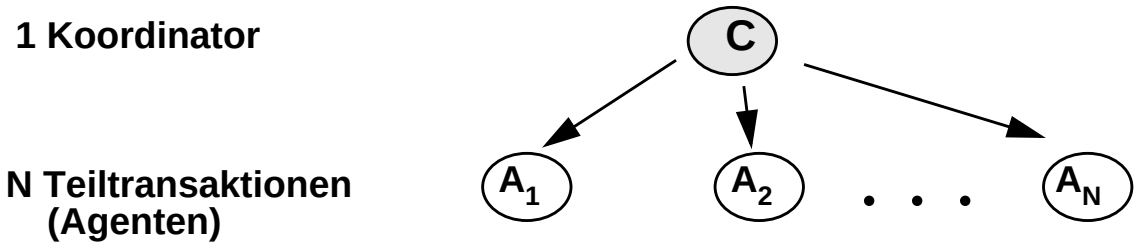
- Verteilung von DB-Operationen (Forts.)
  - Beispiel



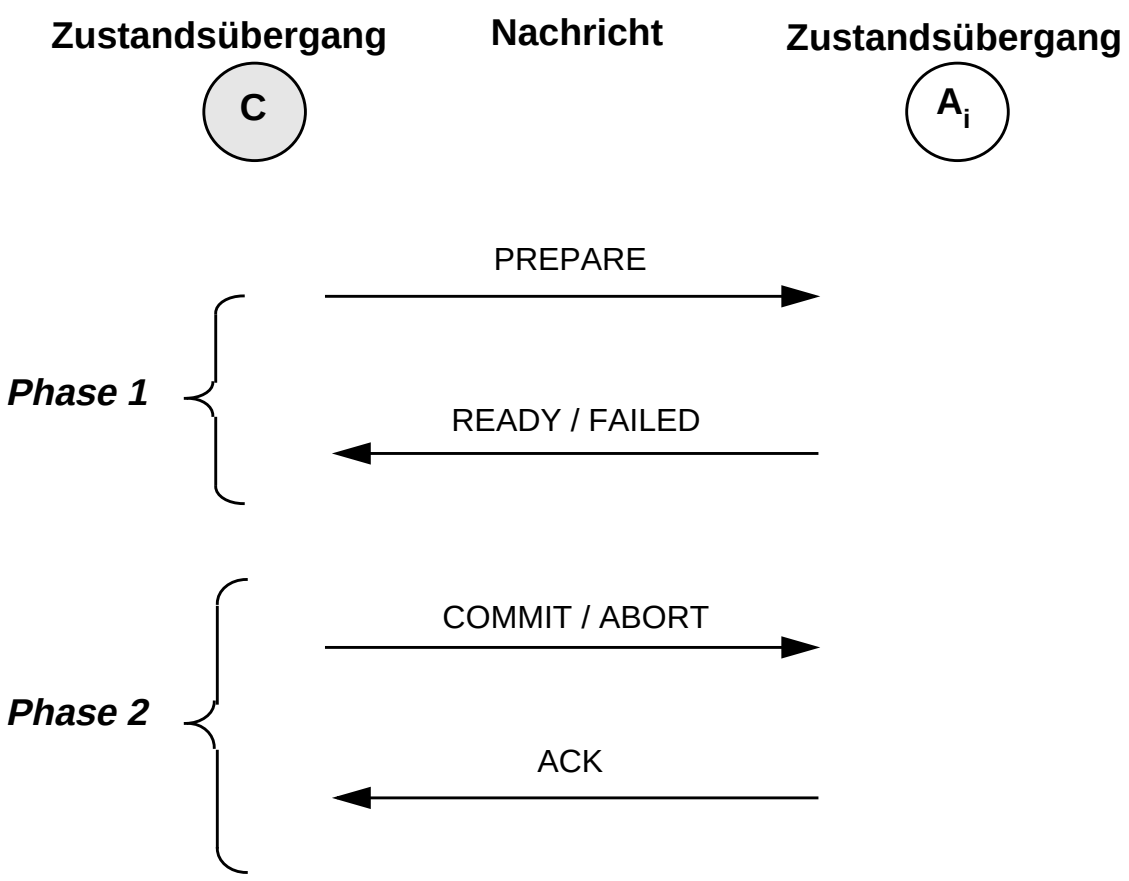
Eingabenachricht (Parameter) lesen
BEGIN WORK
CONNECT TO R1.DB1 ...
UPDATE ACCOUNT
SET BALANCE = BALANCE - :S
WHERE ACCT_NO = :K1;
CONNECT TO R2.DB2 ...
UPDATE GIROKTO
SET KSTAND := KSTAND+ :S
WHERE KNUMMER = :K2;
• • •
COMMIT WORK
DISCONNECT ...
Ausgabenachricht ausgeben

Transaktionssysteme (9)

❏ Verteilte Transaktionsbearbeitung

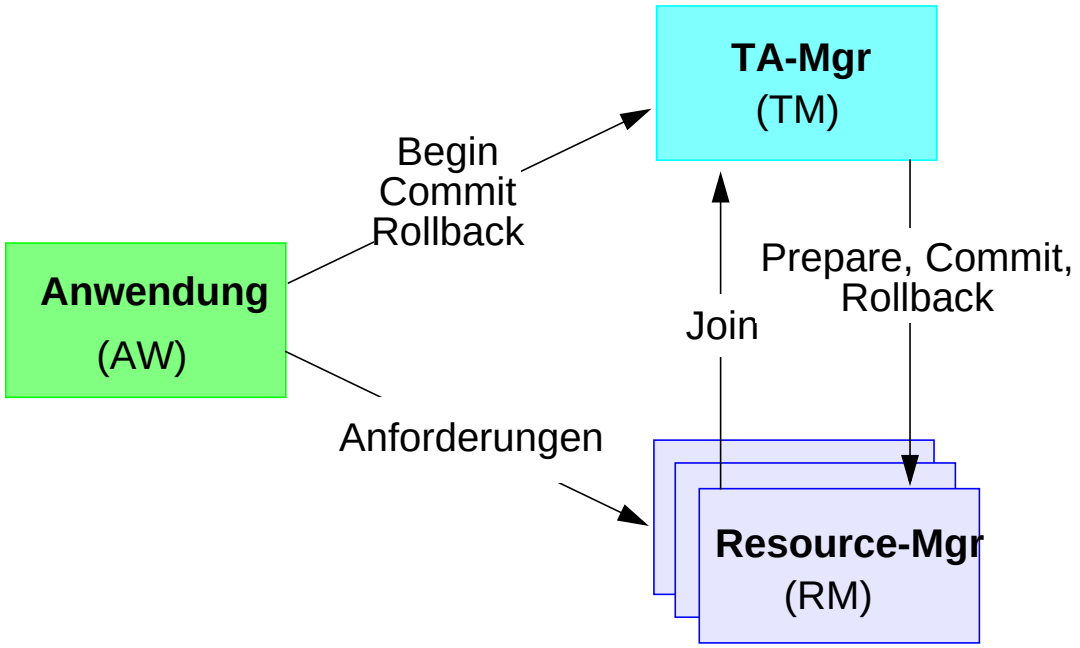


❏ Zweiphasen-Commit-Protokoll



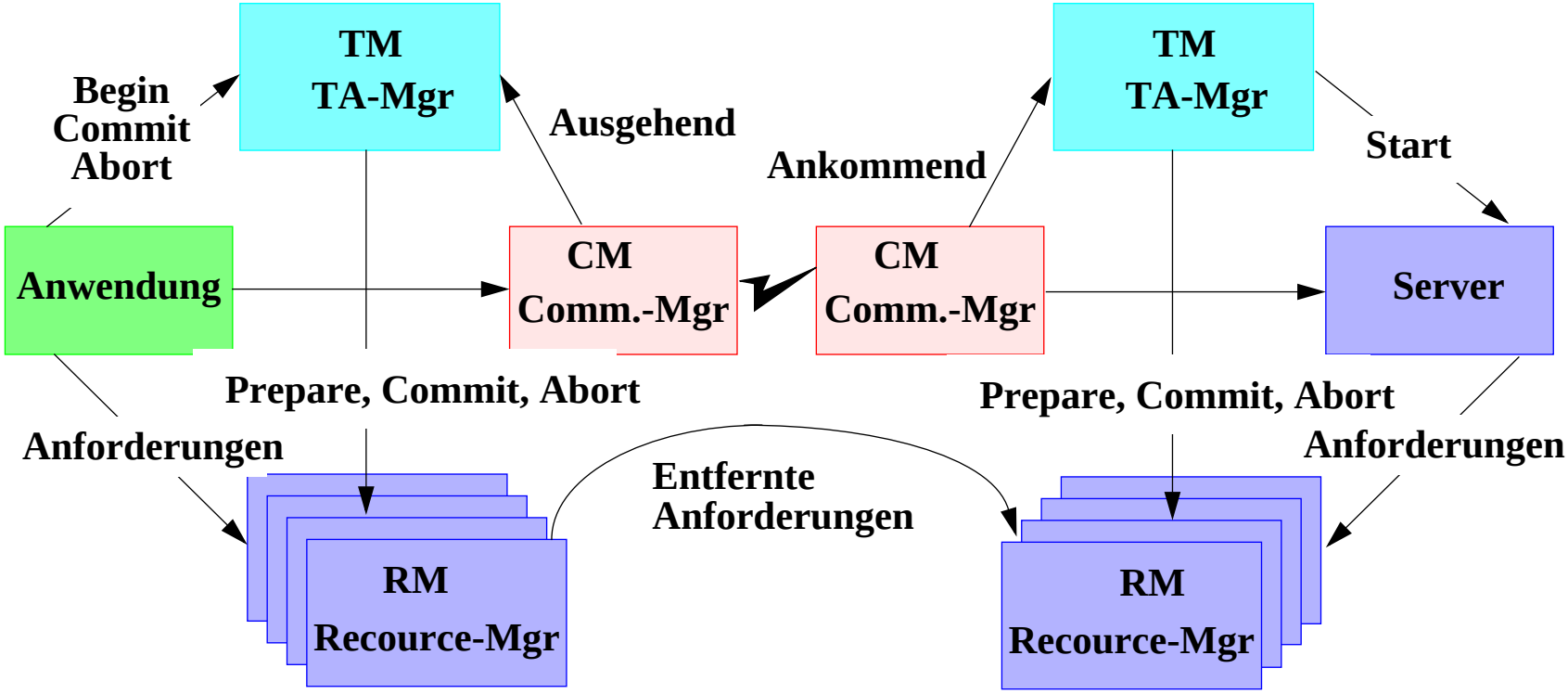
Zusammenspiel der Transaktionsdienste (1)

- Verteilte Transaktionsverwaltung nach X/OPEN DTP (1991)



Zusammenspiel der Transaktionsdienste (2)

- Verteiltes TP-Modell nach X/Open



Zusammenspiel der Transaktionsdienste (3)

- Strikte Unterscheidung zwischen TP-Monitor und TA-Mgr
  - viele Systembenutzer benötigen den TP-Monitor nicht: z. B. Stapel-AW, Ad-hoc-Anfragen über eine direkte SQL-Schnittstelle
  - spezielle AW-Systeme (z. B. CAD) haben ihre eigene Terminalumgebung
  - alle Aktivitäten brauchen jedoch Transaktionsunterstützung
- Eine Konsequenz ist die Trennung der Komponenten zur Durchführung
  - der **Transaktionskontrolle** (TA-Mgr) und
  - des **transaktionsorientierten Betriebsmittel-Scheduling** (TP-Monitor)

Zusammenspiel der Transaktionsdienste (4)

- TP-Monitor-Unterstützung für RM
  - *Start und Restart des Systems*
    - alle RM der spezifizierten Konfiguration müssen zunächst gestartet werden
    - das Recovery-Protokoll bei Crash wird zwischen RM und TA-Mgr direkt abgewickelt
    - der TP-Monitor hat hierbei nur Kontrollfunktion
  - *Definition eines neuen RM*
    - Übernahme der Beschreibungsdaten in den Katalog und Aktualisierung der Konfigurationsdaten
  - *Änderung der Prozeßkonfiguration*
    - z. B. Erweiterung einer Server-Klasse
  - *Handhabung von TRPCs*
    - als grundlegender Mechanismus für die Zusammenarbeit aller Komponenten

TP-Monitor (1)

- Grundlage: Resource Manager (RM)
  - Systemkomponenten, die **Transaktionsschutz für ihre Betriebsmittel** (BM) in der Art bieten, daß diese BM in globale Transaktionsdienste eines TP-Monitors integriert werden können, heißen **Resource Manager**.
- Aufgaben
  - **Bewältigung der Heterogenität**:
    - Zugriff auf heterogene DB oder verschiedenartige RM bei gleichzeitiger Gewährleistung von ACID für die gesamte Funktion
  - **Kontrolle der Kommunikation**:
    - Bei verteilter Verarbeitung unterliegt die Kommunikation auch der TA-Kontrolle. Das kann erreicht werden durch einen Mechanismus wie ‘transactional remote procedure call’ (TRPC).
    - Der Kommunikations-Manager handhabt und kontrolliert Nachrichten und Sitzungen; er ist deshalb ein RM.
  - **Verwaltung der Terminals**:
    - Nachrichtenkommunikation mit unterschiedlichsten Terminals (Workstations, . . ., Zapfsäulen)
    - Nachrichtenkommunikation muß Teil der TA sein, damit für den Endbenutzer die ACID-Eigenschaften gewährleistet werden können.
    - Der TP-Monitor muß diese Aufgabe übernehmen (schwierige Probleme im Fehlerfall)

TP-Monitor (2)

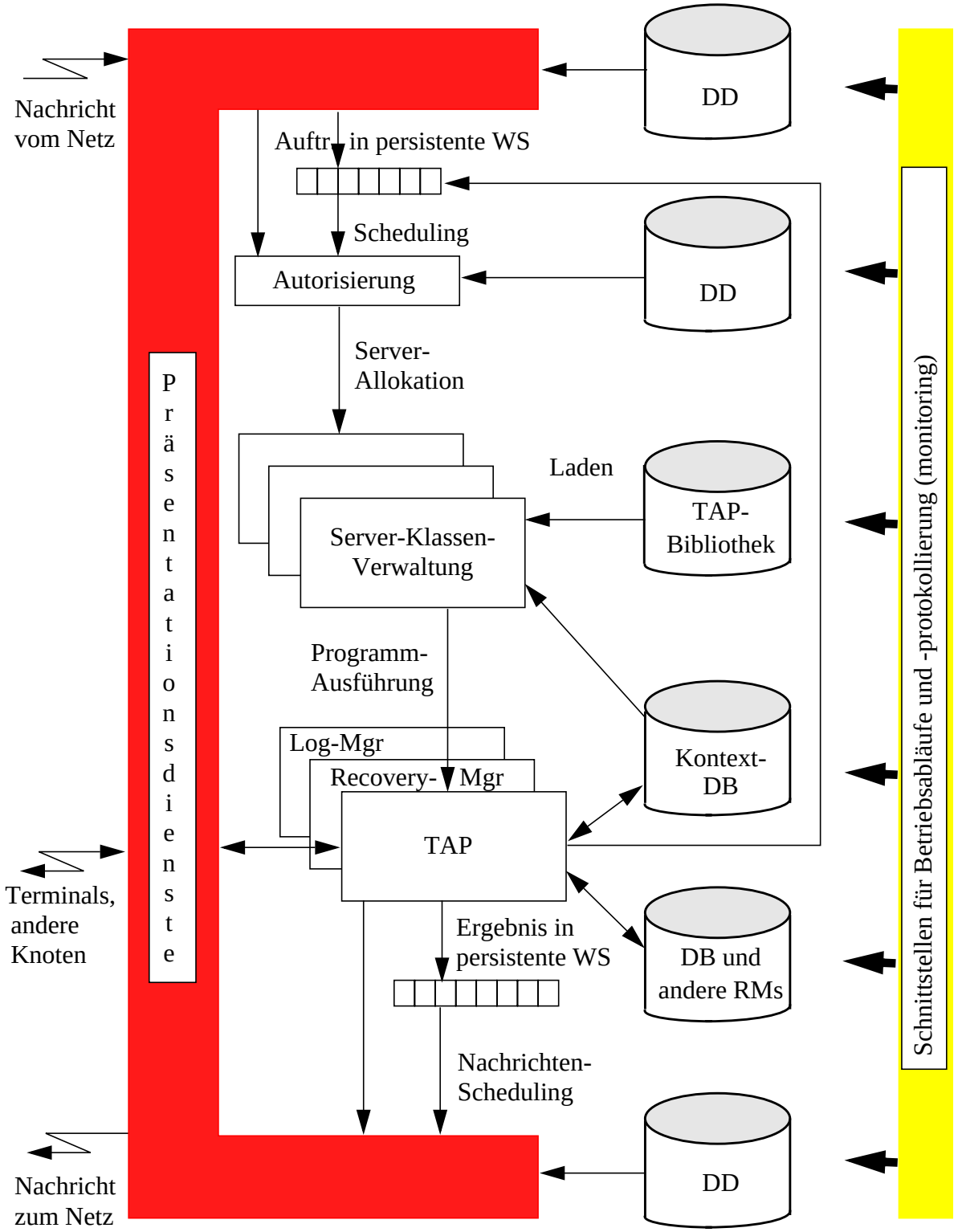
- Aufgaben (Forts.)
  - **Präsentationsdienste**:
    - Um ACID für den Endbenutzer zu realisieren, ist im Fehlerfall seine “Sicht” wiederherzustellen.
    - Wenn er bspw. ein X-Window-System verwendet, sind Fenster, Cursorpositionen u. a. wieder aufzubauen, d. h., ein X-Client muß in einer TA-orientierten Umgebung als RM agieren.
  - **Kontextverwaltung**:
    - Kontexte sind bspw. bei Mehrschritt-TA oder bei Abwicklung einer langen TA durch “Mini-Batches” erforderlich.
    - Die Speicherung und Wiederherstellung von Kontexten ist gebunden an die SoC der TA, die einen Kontext erzeugte oder zuletzt modifizierte.
    - Wartung der Kontexte ist Aufgabe des TP-Monitors.
  - **Start/Restart**:
    - Praktisch alle Komponenten einer TA-orientierten Ablaufumgebung benötigen Transaktionsdienste.
    - Deshalb muß der TP-Monitor auch den Restart nach einem Fehler übernehmen, um einen konsistenten Zustand nach den ACID-Regeln aufzubauen.

TP-Monitor (3)

- Aufgaben (Forts.)
  - **Programmverwaltung**
    (siehe oben: TP-Monitor-Unterstützung für RM)
  - **Konfigurationsverwaltung**
    (siehe oben: TP-Monitor-Unterstützung für RM)
  - **Lastbalancierung**
  - **Autorisierung**
  - **Bereitstellung von Administrationsschnittstellen**

TP-Monitor (4)

❑ Kontrollfluß durch einen TP-Monitor (vereinfacht)



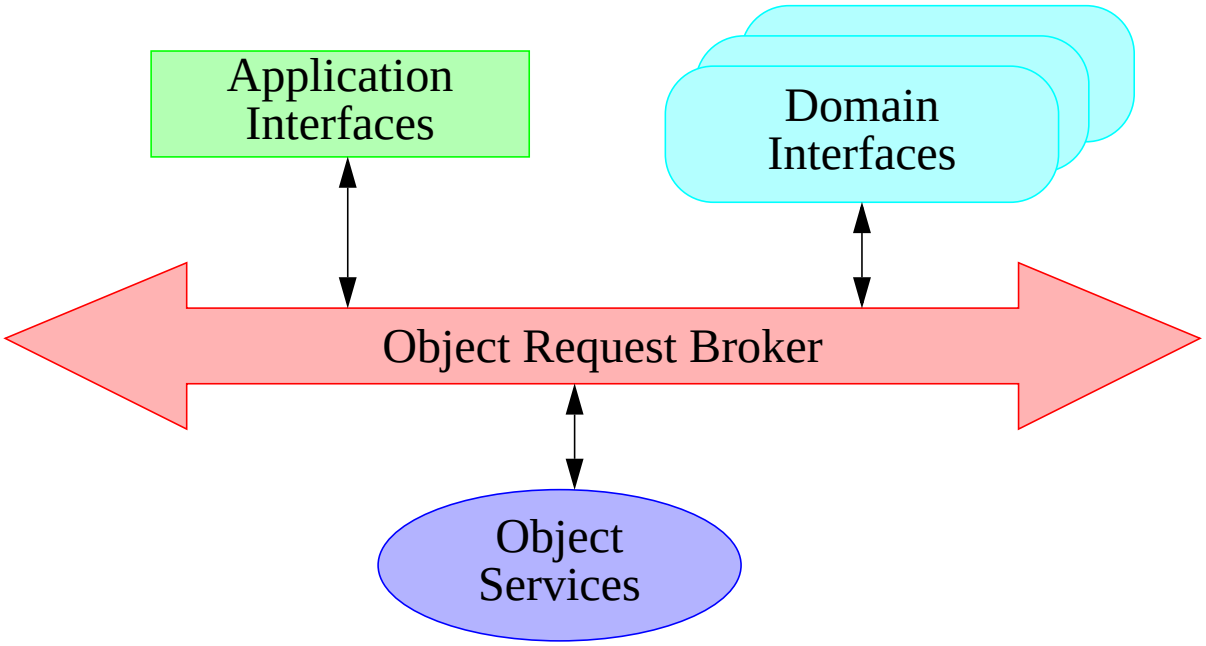
TP-Monitor (5)

❑ Middleware-Aspekt

- ➪ TA-Konzept als Basis für (DB-)Middleware
  - TA-Schutz für verteilte Abläufe notwendig
  - damit sind die Prinzipien der Verteilten Transaktionsverwaltung grundlegend für jede Art der Middleware-Integration
  - TP-Monitor/TA-Manager bzw. proprietäre Komponente, die Transaktionskontrolle durchführt, als notwendige Komponente jeder DB-Middleware
- ➪ TP-Monitor selbst gehört zur DB-Middleware
  - Integration von verteilten, heterogenen RM
  - Herstellung von globalen Sichten
    (TA-Routing, Vert. Progr., Vert. v. DB-Ops.)

TA in der Middleware - CORBA (1)

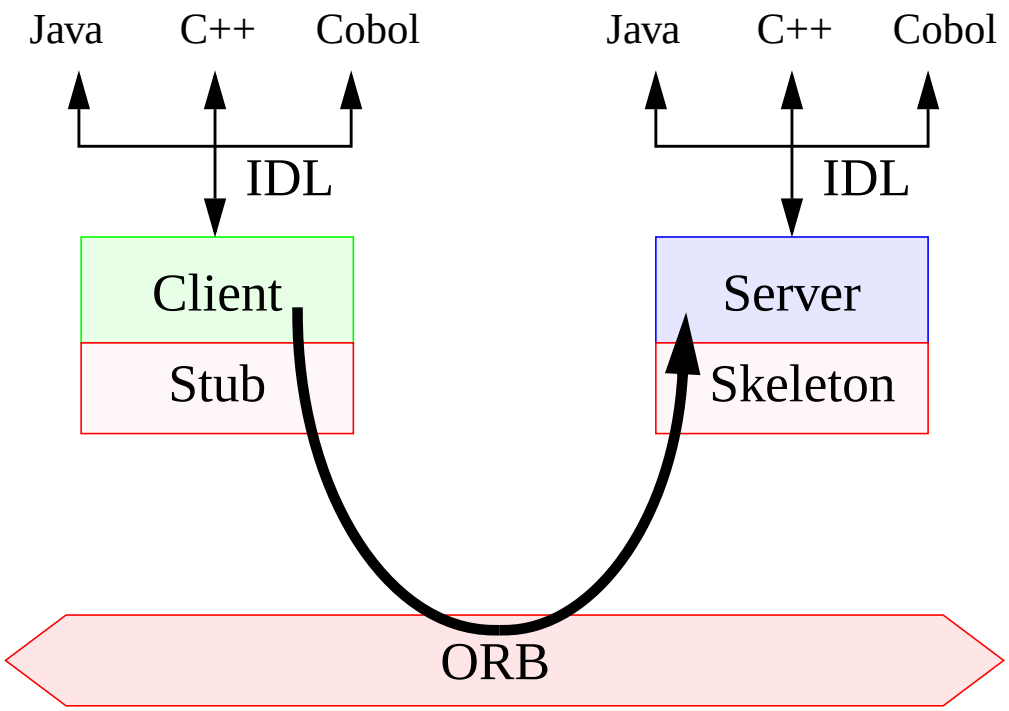
- Beispiel: CORBA
- Object Management Architecture (OMA)



- Interfaces in unterschiedlichen Kategorien
  - Object Services (horizontal)
  - Domain Interfaces (vertikal)
    - Telekommunikation
    - Finanzdienstleistungen
    - E-Commerce
    - Medizin
    - ...
  - Application Interfaces

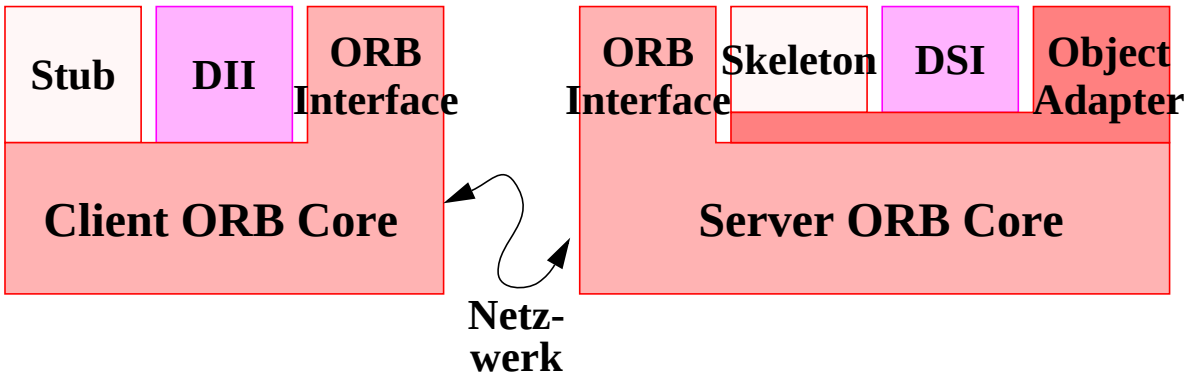
CORBA (2)

- Object Request Broker (ORB)
  - Vermittlung der Dienstaufrufe zwischen verschiedenen Interfaces



CORBA (3)

- CORBA-Kernkomponenten
  - Objektreferenzen (Interoperable Object References, IOR)
  - Object Request Broker (ORB)
  - Objektadapter
  - Stubs und Skeletons
  - Dynamic Invocation/Skeleton Interface (DII/DSI)



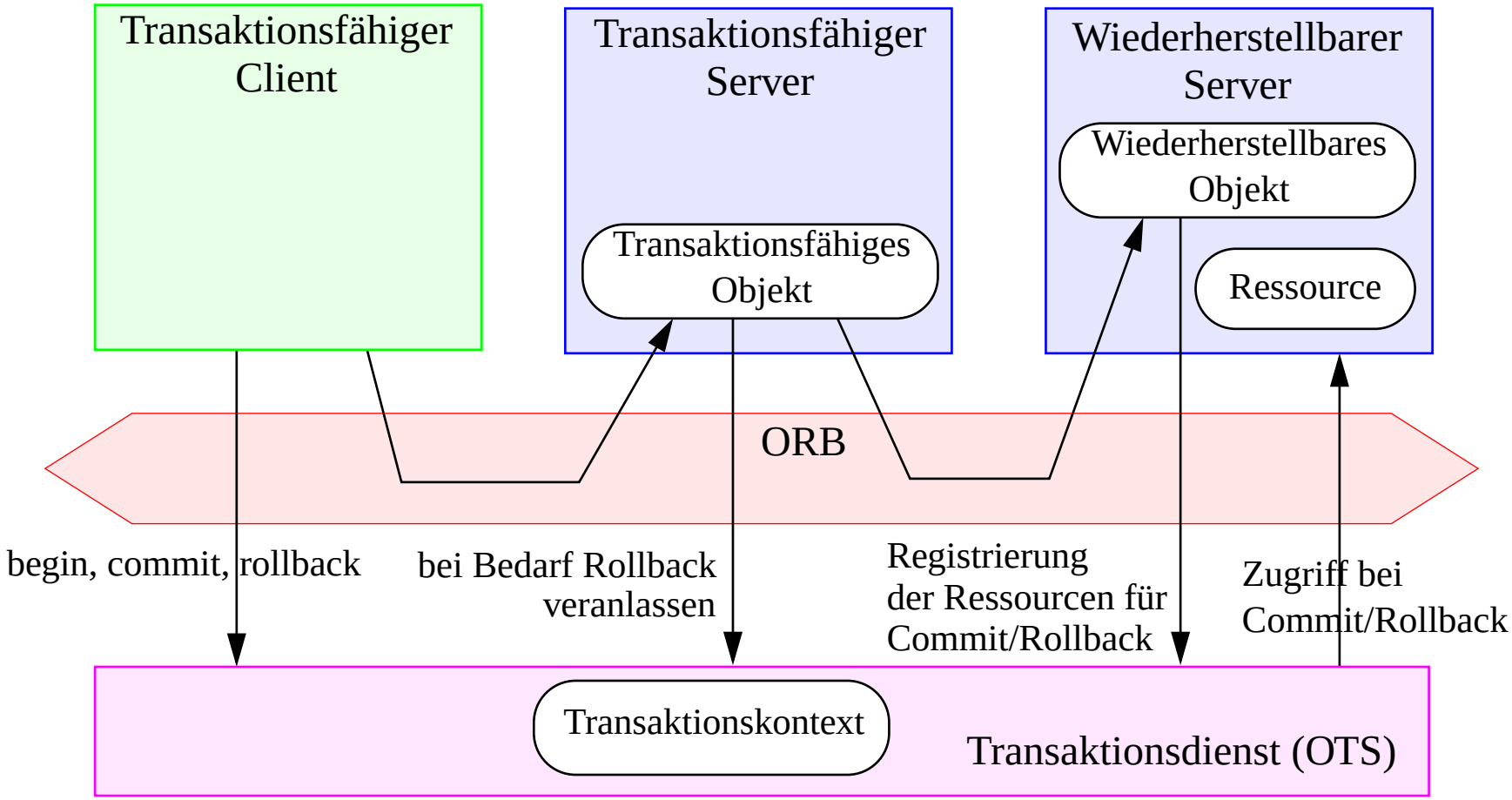
  - Dienst-spezifisch
    - Stub
    - Skeleton
  - identisch für alle Anwendungen
    - ORB Interface
    - DII
    - DSI

CORBA (4)

- CORBA-Service: Transactions
  - flache, geschachtelte Transaktionen
  - X/OPEN DTP
  - Aufgaben des Object Transaction Service (OTS) entsprechen im wesentlichen denen des TA-Managers im X/OPEN-Modell

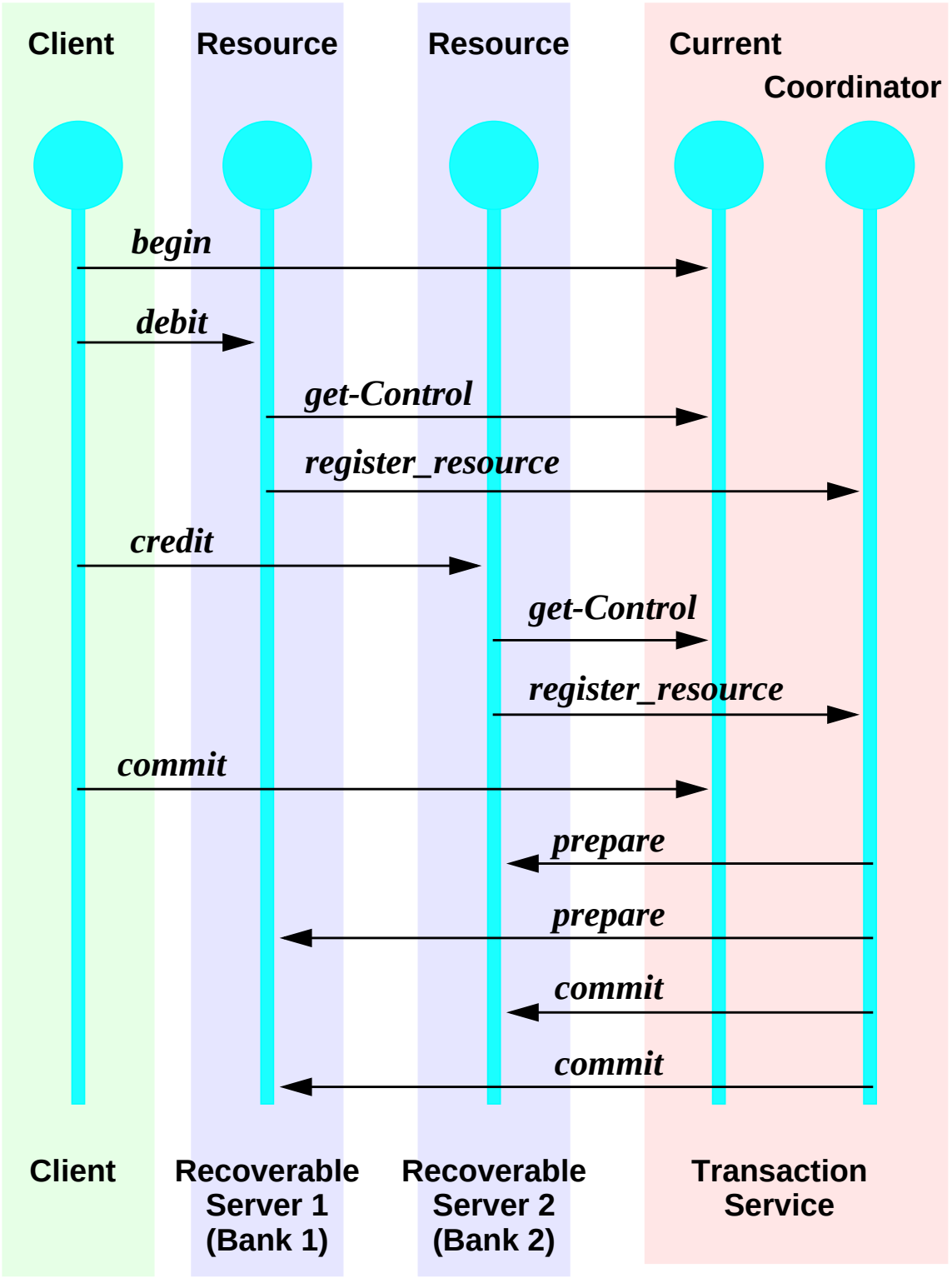
CORBA (5)

- Object Transaction Service (OTS)



CORBA (6)

- OTS - Schnittstellen



CORBA (8)

- CORBA-Service: Concurrency
  - Interfaces zum Anfordern und Freigeben von Sperren auf *shared resources*
  - falls im Dienst eines transaktionalen Clients: Freigabe von Sperren wird durch den OTS veranlaßt
  - falls im Dienst eines nicht-transaktionalen Clients: Verantwortlichkeit für die Sperrfreigabe bei Client
  - Serialisierbarkeit
  - R/X/Intention/U-Sperren
  - Methode *try_lock* gibt Kontrolle zurück, statt zu blokkieren
  - Locksets
    - Menge von Sperren , die später als Einheit freigegeben werden sollen
    - mehrere Locksets können miteinander assoziiert werden
  - Lock-Coordinator
    - koordinierte Freigabe von Locksets
    - wird vom OTS genutzt

Zusammenfassung

- Wiederholung:
  - Transaktionen im zentralen Fall
  - Verteilte Transaktionssysteme / TP-Monitore
  - Transaktionen in der Middleware
- Ausblick