Kapitel 5

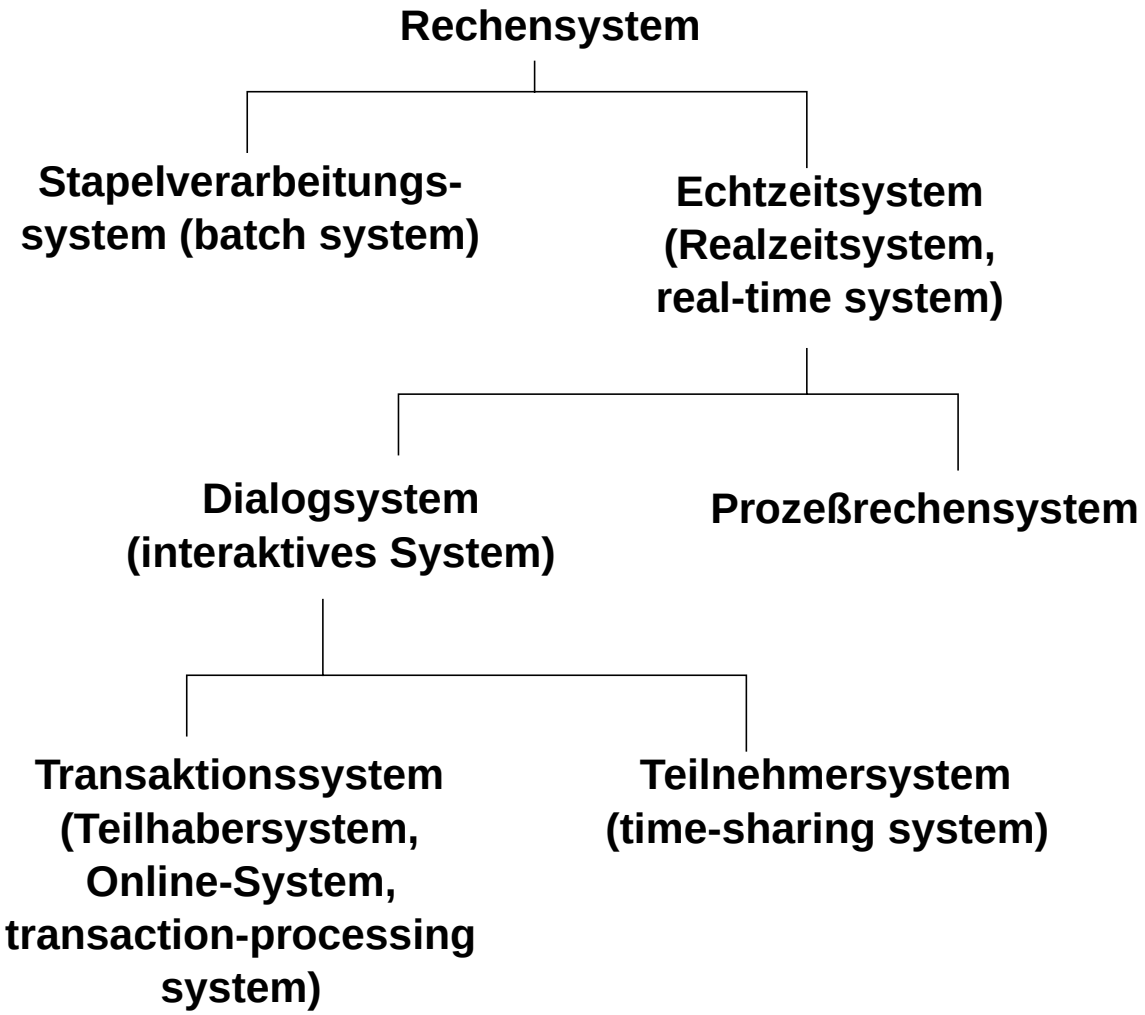
TP-Monitore

Inhalt

- ❏ Motivation
- ❏ Anwendungen und Anforderungen
- ❏ TP-Monitor-Architektur
- ❏ Präsentation
- ❏ Workflow-Kontrolle
- ❏ Transaktionsprogramme
- ❏ Betriebssystem-Prozesse
- ❏ DB-Server vs. TP-Monitore
- ❏ Transaktionen im Internet
- ❏ Zusammenfassung

Motivation (1)

❏ Systemklassen



➪ Stapelverarbeitungssystem:

- Optimierungsziel: hohe Auslastung
- Sammlung der Aufträge verschiedener Benutzer in einem Stapel; Bearbeitung nach Priorität
- weiche Zeitrestriktionen; lange Antwortzeiten
- vollständige Formulierung umfangreicher Aufträge

Motivation (2)

- Systemklassen (Forts.)
  - Echtzeitsystem:
    - Optimierungsziel: hohe Verfügbarkeit
    - Bearbeitung der Aufträge *sofort*
    - harte Zeitrestriktionen: kurze Antwortzeit
    - viele kleine Aufträge
  - Prozeßrechensystem:
    - Kommunikationspartner („Benutzer“): technischer Prozeß
    - Austausch kompakter, redundanzfreier Daten
    - extrem harte Zeitrestriktionen: Antwort muß innerhalb vorgegebener Zeitspanne erfolgen, sonst sinnlos
  - Dialogsystem:
    - Kommunikationspartner (Benutzer): Mensch
    - Präsentation von Information mit Erläuterungen (Masken, Farben, Graphiken)
  - Teilnehmersystem:
    - mehrere Benutzer gleichzeitig
    - jeder ein eigenes universelles, virtuelles Rechensystem
    - Funktionen: Systemkommandos
    - Umgebung für Programmierer

Motivation (3)

- Systemklassen (Forts.)
  - Transaktionssystem (Teilhabersystem):
    - sehr viele Benutzer gleichzeitig
    - anwendungsspezifische, problemorientierte Funktionen
    - für EDV-Laien
    - gemeinsamer Datenbestand
- Typische Anwendungen für Transaktionssysteme (TP-Systeme):
  - Platzbuchung
  - Transportsteuerung
  - Produktionsplanung
  - Auftragsabwicklung
  - Girokontenführung
  - Personalverwaltung
  - ...
- Kenngrößen eines Flugbuchungssystems:
  - 3000 Terminals
  - 5 Mrd. Bytes Plattenspeicher
  - 50 Buchungen / s

Motivation (4)

- Beispiel eines großen Transaktionssystems: TWA-Reservierungssystem
  - Aufgaben:
    - Platzbuchung in Flügen der TWA und anderer Gesellschaften
    - Tickets ausgeben
    - Gepäcküberwachung
    - Frachtübernahme (Cargo)
    - Flugzeuggewicht und -balance
    - Kreditkartenprüfung
    - ...
  - 50% der Buchungen von fremden Reisebüros
  - **Last** (Stand 1983):
    - 7 Millionen Transaktionen pro Tag
    - Spitzendurchsatz ≈ 200 TA/s
    - angestrebt: 90% der Antworten in 3 s,
    - mittlere Antwortzeit 1,5 s
  - Systemkonfiguration:
    - Zentralrechner IBM 9083 (12 MB Hauptspeicher, ≈ 7,4 MIPS) exklusiv genutzt (Entwicklung auf separatem Rechner)
    - 144 Plattengeräte à 371 MB (72 mit den Daten, die anderen als Spiegelplatten)
    - 11000 – 12000 Terminals weltweit

Motivation (5)

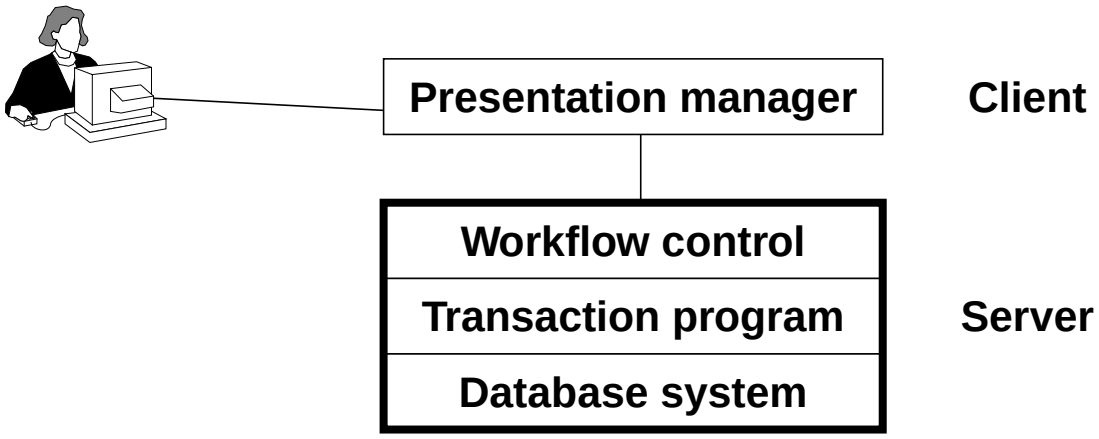
- Bestandteile eines Transaktionssystems
  - Hardware:
    - Terminals
    - Datenkommunikationsnetze
    - Verarbeitungsrechner/Hosts mit Plattenperipherie
  - Software:
    - Netz-Software/Kommunikationssoftware)
    - Betriebssystem
    - Transaktionsmonitor (TP-Monitor)
    - Datenbankverwaltungssystem (DBVS)
    - Anwendungsprogramme/Transaktionsprogramme
  - **Benutzer** (Anwender):
    - Endbenutzer (Benutzer im engeren Sinne)
    - Programmierer
    - Administrator

Motivation (6)

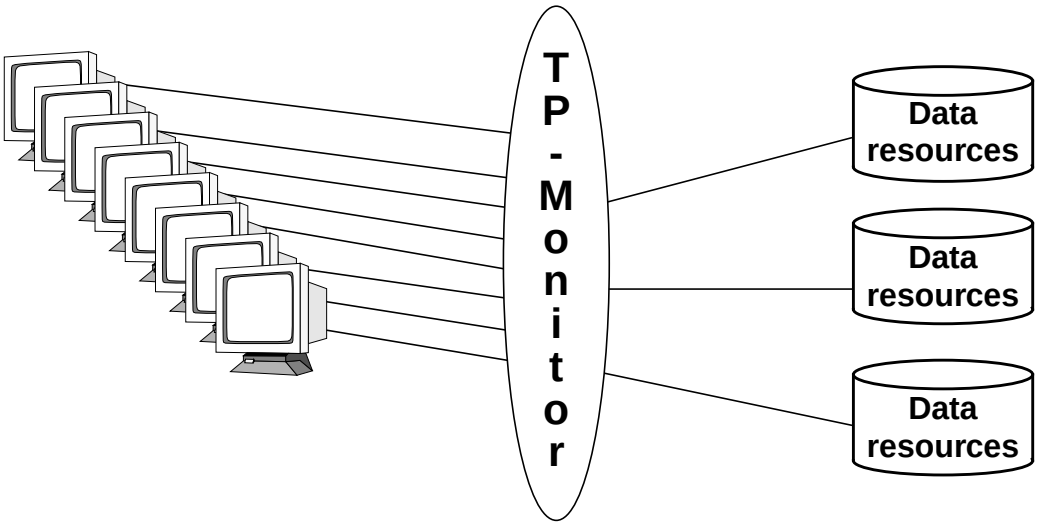
- Anforderungen an die Realisierung
  - große Anzahl von angeschlossenen Terminals (10^2 - 10^4)
  - konkurrierende Ausführung vieler Funktionen
  - Zugriff auf gemeinsame Daten mit
    - größtmöglicher Aktualität
    - Erhaltung der Konsistenz
    - Zugriffskontrolle
  - hohe Verfügbarkeit
    - kurze Antwortzeiten
    - Ausfallsicherheit
  - flexible Reaktion auf stochastisches Verkehrsaufkommen
  - Unterstützung der Administration bei:
    - Systeminstallation
    - Änderungen und Ergänzungen
    - Leistungsüberwachung und Tuning

TP-Monitor-Architektur (1)

- TP-System
  - Client/Server-Sicht

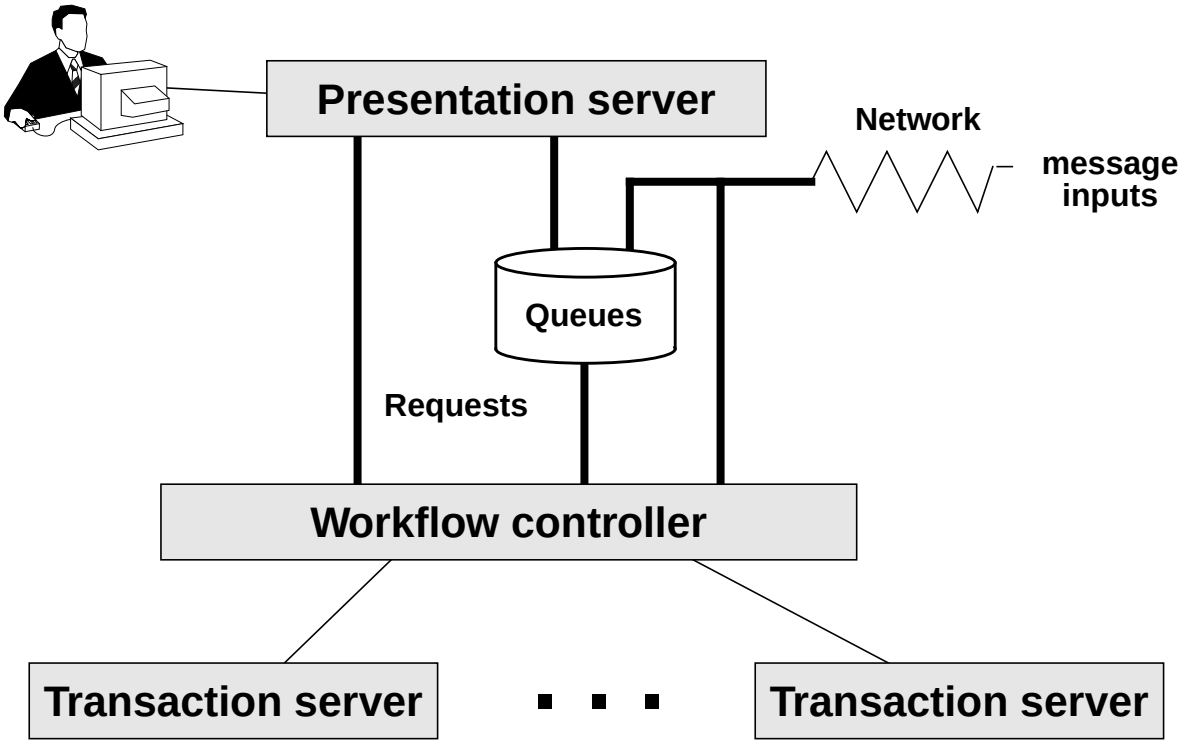


  - TP-Monitor-Einbindung

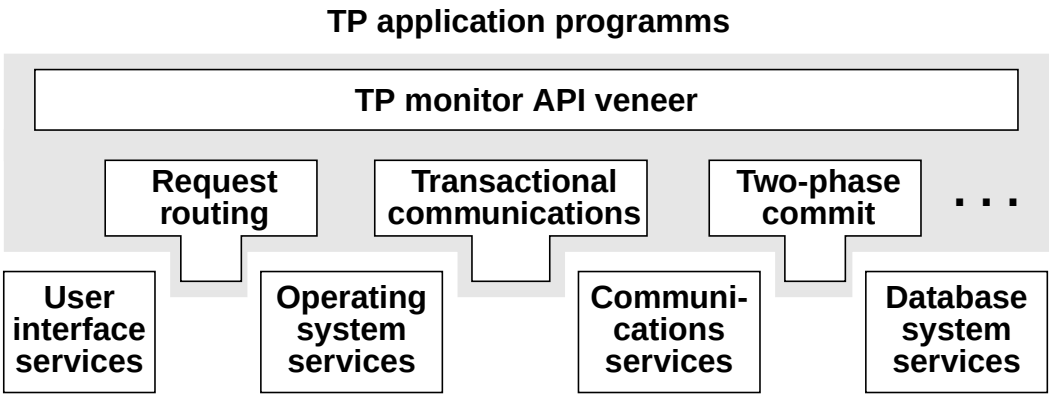


TP-Monitor-Architektur (2)

- TP-Monitor-Komponenten
  - 3-Tier-Modell

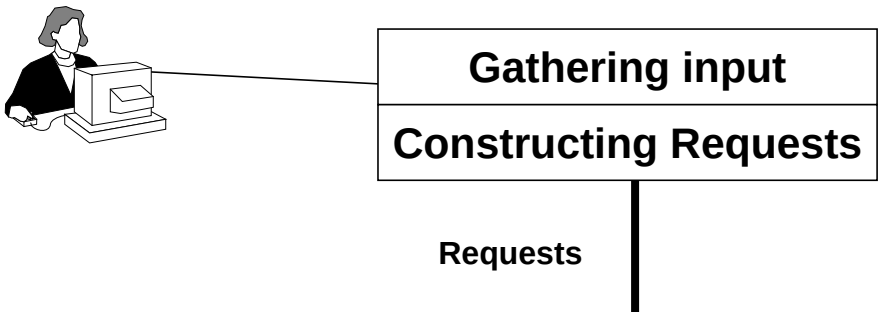


  - TP-Monitor: “Klebstoff und Furnier”

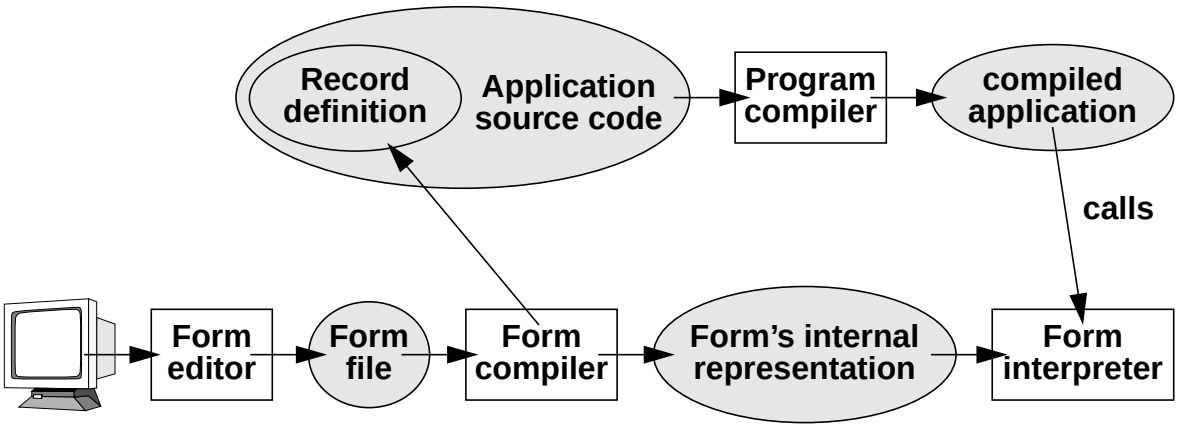


Präsentation (1)

- Presentation Server
  - Layers



  - Forms Manager Tools



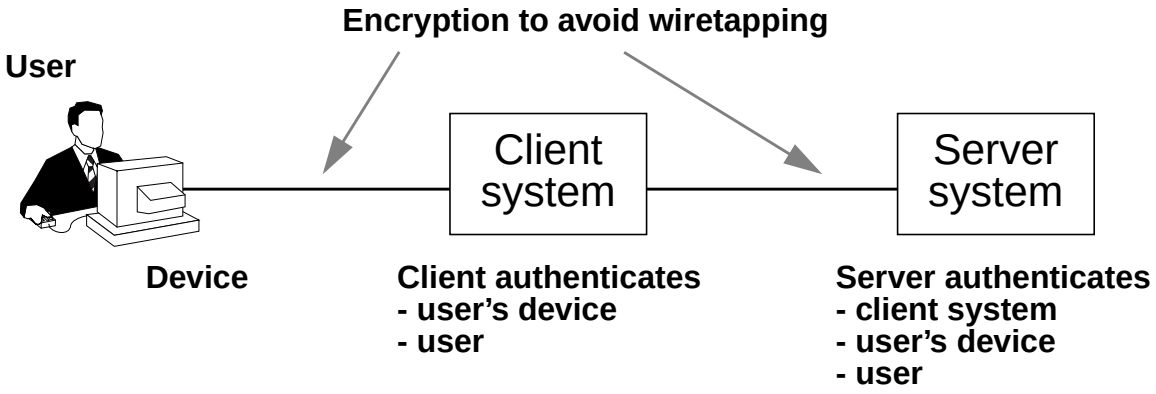
  - Request-Format

User Name	Device ID	Request Type	Request-Specific Parameters

    - Device-ID erlaubt Rückschlüsse auf Device-Typ; z. B. PC mit speziellem BS, speziellen Kommunikationsprotokollellen, etc.
    - vgl. auch RPC-Begriff, Kapitel 6

Präsentation (2)

- Presentation Server (Forts.)
  - Authentifizierung
    - Bestimmung der Identität von User und Display-Device, üblicherweise über Passwords
    - noch trauen TP-Monitore der Client-Authentifizierung nicht, sondern nutzen eigene Komponente
    - es zeichnet sich jedoch ab, dass mit der Entwicklung von Komponenten-Infrastrukturen deren Authentifizierungsmechanismen ausreichend vertrauenswürdig werden
    - jede Client-Nachricht an den Workflow-Controller enthält authentifizierte(n) User-Name und Device-ID



    - geographic entitlement: Berechtigungen beziehen sich auf konkrete Device-IDs; Device-Typ nicht ausreichend; z. B., Zugang zu Stock-Exchange nur aus Trading-Room;
    - zeitliche Beschränkungen möglich; z. B. Zugriff nur zu bestimmten Zeiten
    - Presentation-Server muß Administrations-Schnittstelle bereitstellen; z. B. Einstellungen, Angabe von Rollen, Passwords, etc.

Präsentation (3)

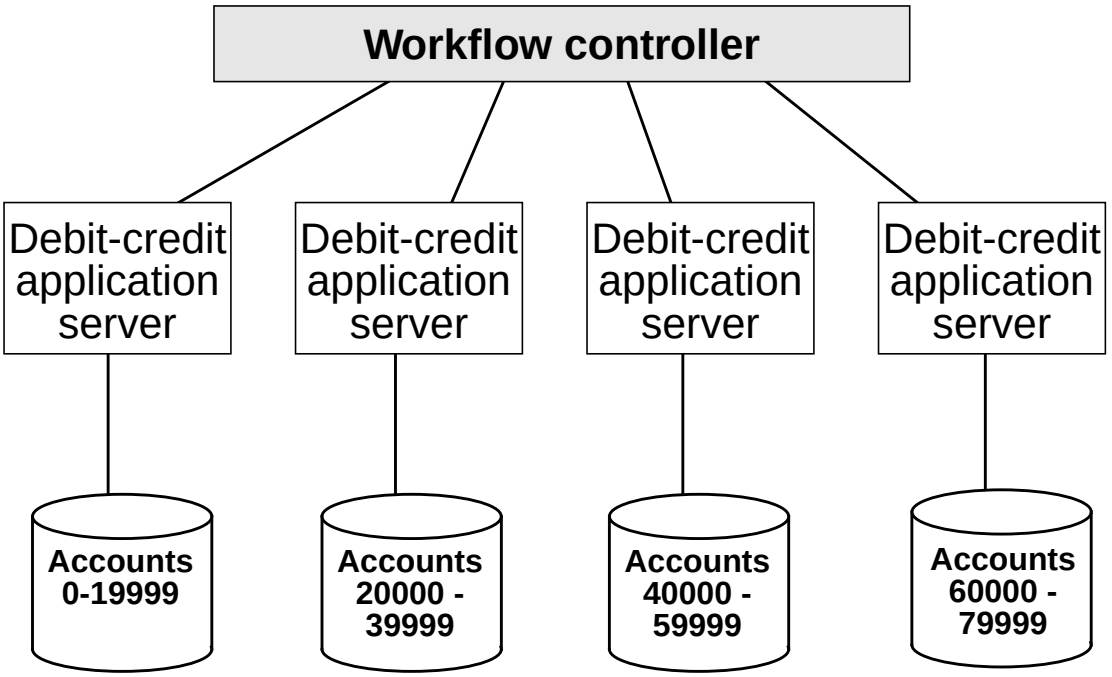
- Presentation Server (Forts.)
  - Communication
    - Abbildung interner Bezeichnungen (Device-Namen, Meldungen, Parameter, etc.) auf logische Bezeichner, so daß Programmierer semantisch bedeutungsvolle Namen in ihren Nachrichtenspezifikationen nutzen können
  - Logging
    - von Nachrichten zum Zwecke der Fehler-Rückverfolgung
    - manchmal übernehmen dies auch die Endgeräte (Display-Devices); Beispiel: Geldautomaten

Workflow-Kontrolle (1)

- Wesentliche Aufgaben des Workflow-Controllers
  - Abbildung von Requests auf Server (Routing)
  - ggf. “Klammern” von Transaktionen
  - Rückgabe der Ergebnisse an Presentation-Server
- Routing
  - Directory-Services
    - Verwaltung von Einträgen zur Abbildung von logischen Server-Namen in Server-IDs
    - dynamische Funktion aufgrund periodischer Re-Konfigurierung des TP-Monitors; z. B., Lokationen von Servern können sich ändern nach Ausfall eines Knotens bzw. bei Verteilung zu Zwecken der Lastbalancierung
    - es zeichnet sich ab, dass TP-Monitore netzwerkweite Directory-Server, wie sie z. B. durch Komponenteninfrastrukturen zur Verfügung gestellt werden, nutzen werden
    - bisher jedoch meistens eigene Directory-Server der TP-Monitore

Workflow-Kontrolle (2)

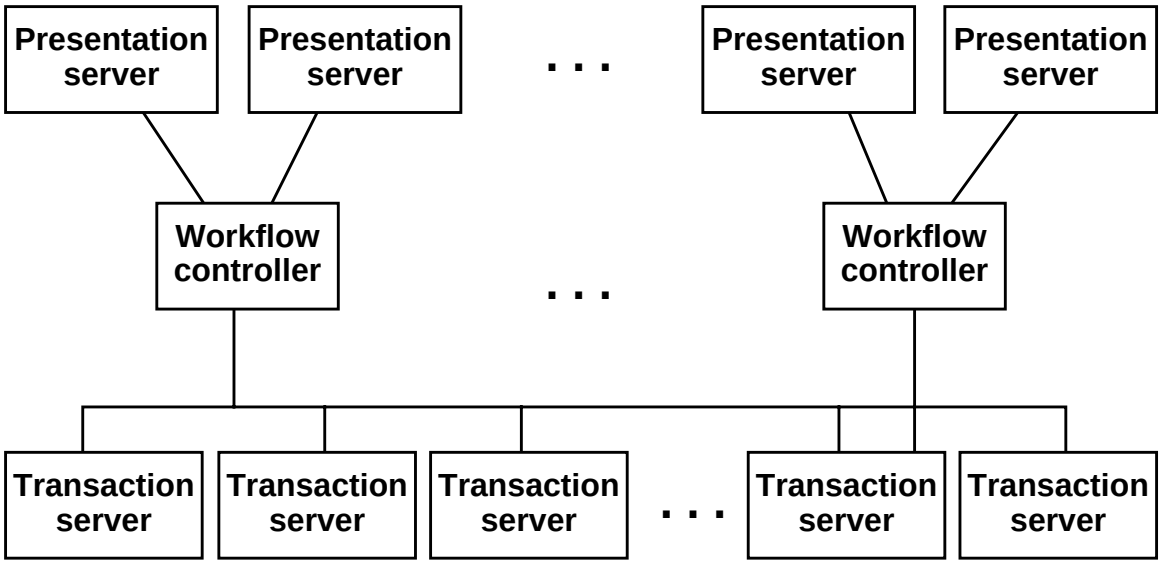
- Routing (Forts.)
  - Parameter-based Routing



  - Sessions
    - Verwaltung des Zustands der Kommunikation zwischen zwei Partnern
    - während einer Session wird über Nachrichten kommuniziert, die jeweils den Session-Zustand beinhalten
    - beinhaltet in der Regel auch Sicherheitsinformationen
      - oft per-message-security-checking notwendig; z. B. wenn ein Benutzer nur bestimmte Request-Typen in Anspruch nehmen darf
      - kann zwischen Workflow-Controller und Transaction-Server vereinfacht werden, wenn Workflow-Controller als Trusted-Client angesehen werden kann

Workflow-Kontrolle (3)

- Routing (Forts.)
  - Minimierung der Anzahl von Sessions (Multilevel Routing)



- Minimierung des Aufwands
- Skalierbarkeit
- Fehlertoleranz
- Gruppierung oft nach geografischen Gesichtspunkten

Workflow-Kontrolle (4)

- Workflow-Logik
  - Abbildung von Requests auf Transaction-Server (Transaktionen) und Kontrolle des zugehörigen Flow-of-Work
  - Möglichkeiten
    - eigene Sprache
    - Funktionsbibliothek
    - Delegation der Kontrolle von Mehrschrittprogrammen auf die Transaction-Server-Ebene
      - Root-Pogramm sinnvoll, das nur die Kontrolle übernimmt
    - Unterstützung von Parallelität
      - Encina[1] (Transarc)
      - STDL (structured transaction definition language) von ACMSxp[1] (Digital)

1. siehe Systembeispiele in nachfolgendem Kapitel

Workflow-Kontrolle (5)

- "Klammern" von Transaktionen
  - Workflow-Controller 'verpackt' in Zusammenarbeit mit einem Transaktions-Manager, falls notwendig, mehrere Transaction-Server-Aufrufe in eine übergeordnete Transaktion
  - Request Message Integrity
    - bei Einlesen des Requests innerhalb der Transaktion wird im Falle eines Aborts dieses Einlesen rückgängig gemacht und damit kann der Request von einer neuen Transaktion wieder aufgegriffen werden

```
// Example A
Get-input-request;
Start;
  ...
Commit;
```

```
// Example B
Start;
  Get-input-request;
  ...
Commit;
```

  - Pull vs. Push

```
void workflowController
      {  Start;
         Get-input-request (RequestMsg* req);
         ...
      }
```

```
void workflowController (RequestMsg* req)
      {  Start;
         ...
      }
```

    - im Push-Fall explizite Accept-Operationen zum Einlesen der Input-Parameter zwecks Vermeidung des o.a. Request-Message-Integrity-Problems sinnvoll
    - vgl. Kapitel 6

Workflow-Kontrolle (6)

- "Klammern" von Transaktionen (Forts.)
  - meistens implizite Klammerung
  - im Falle, dass die TP-Monitor-Sprache ein explizites Setzen der Transaktionsklammern erlaubt, gib es zwei Möglichkeiten:
    - chained
      - 'Alles' innerhalb einer Transaktion
      - Spezifikation von Paaren; nach dem Commit einer Transaktion wird sofort die nächste gestartet bezw. die gesamte Request-Bearbeitung beendet
      - wird von den meisten TP-Monitoren genutzt
    - unchained
      - zwischen zwei Transaktionen sind Aktionen ohne Transaktionsschutz erlaubt
      - flexibler aber 'gefährlicher'

Workflow-Kontrolle (7)

❑ Exception Handling

➪ Transaction Exceptions

- Problem: Behandlung von Transaktionsfehlern (‘unsolicited aborts’) und Systemfehlern
- Exception Handler
  - Was ist zu tun, wenn eine Transaktion einen Fehler verursacht?
  - Wie ist nach der Recovery-Phase weiter zu verfahren?
  - Voraussetzung: (persistente) Statusinformation
  - insbesondere wichtig: Behandlung von ‘unrecoverable resources’
  - beste Vorgehensweise: automatische Verzweigung zum Exception-Handler und Abwicklung des Exception-Handlings in einer eigenen Transaktion (siehe chained-Modell)

➪ (Persistent) Savepoints

```
void Application
    {  Start;
       get-input-request;
       Savepoint (“B”);
       // do some more work
       Commit;
    }
```

```
void ExceptionHandlingForApplication
    {  Restore (“B”);
       // generate diagnostic
       Commit
    }
```

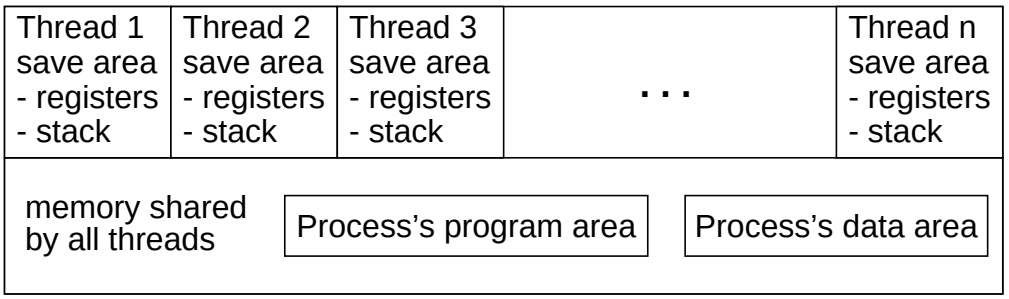
- persistente Savepoints könnten Exception-Handler ersetzen
- werden aufgrund von Realsierungsschwierigkeiten in der Regel (sowohl von DBMS als auch von TP-Monitoren) nicht angeboten

Transaktionsprogramme

- Transaction Server: Applikationsprogramme
  - verrichten die eigentliche Arbeit
  - können selbst wieder verteilt sein
  - Portabilität zwischen verschiedenen TP-Monitoren in der Regel möglich, wenn mit Hilfe von Standard-Programmier- und DB-Sprachen realisiert
  - Wichtigste Anforderungen
    - Skalierbarkeit zu hohen Request-Raten
    - auch Kommunikationsvorgänge müssen transaktional sein
- Wesentliche Aspekte, die sich nicht nur auf Transaction-Server, sondern auch auf die anderen Komponenten der TP-Monitor-Architektur bezeiehen:
  - Geeignete Abbildung auf Betriebssystemprozesse (siehe unten “Betriebssystem-Prozesse”))
  - Geeignete Behandlung von Nachrichten zwischen den Programmen (siehe Kapitel 6)
    - Transaktionskontext muss bei Aufrufen propagiert werden
    - Koordiniertes Beenden der beteiligten Transaktionen (2PC)

Betriebssystem-Prozesse (1)

- Threads
  - Multi-Threaded-Processes



- geringe Anzahl von Prozessen
- geringe Anzahl von Context-Switches
- Gefahr: geringer Schutz zwischen Threads
- dynamische vs. statische Allokation von Threads

  - TP-Monitor-Threads
    - Implementierung der Threads durch den TP-Monitor
    - Problem: synchrone I/O-Operationen einzelner Transaction-Server können dazu führen, dass der gesamte TP-Monitor-Prozess vom BS "schlafen gelegt wird"
    - Konsequenz: der TP-Monitor muss alle diese synchronen I/Os abfangen und die Kommunikation asynchron gestalten
    - Beispiele:
      - ACMS (Digital) und Pathway/TS (Tandem) nutzen Multi-Threading in Presentation-Server und Workflow-Controller-Functions
      - dies war der eigentliche Grund für die Einführung separater Workflow-Sprachen (nur Kontrollfluss- und Kommunikationsfunktionen)

Betriebssystem-Prozesse (2)

- Threads (Forts.)
  - Betriebssystem-Threads
    - Implementierung der Threads durch das Betriebssystem
    - Vorteile
      - synchrone I/Os können lediglich zum “Schlafen-legen” einzelner Threads nicht ganzer Prozesse führen
      - keine Beeinflussung von TP-Monitor- und BS-Scheduling
      - in Multi-Prozessor-Systemen können Threads parallel ausgeführt werden
    - Nachteile
      - höherer Aufwand beim Umschalten zwischen Threads
  - im Allg. wird die Nutzung von Betriebssystem-Threads der von TP-Monitor-Threads vorgezogen

Betriebssystem-Prozesse (3)

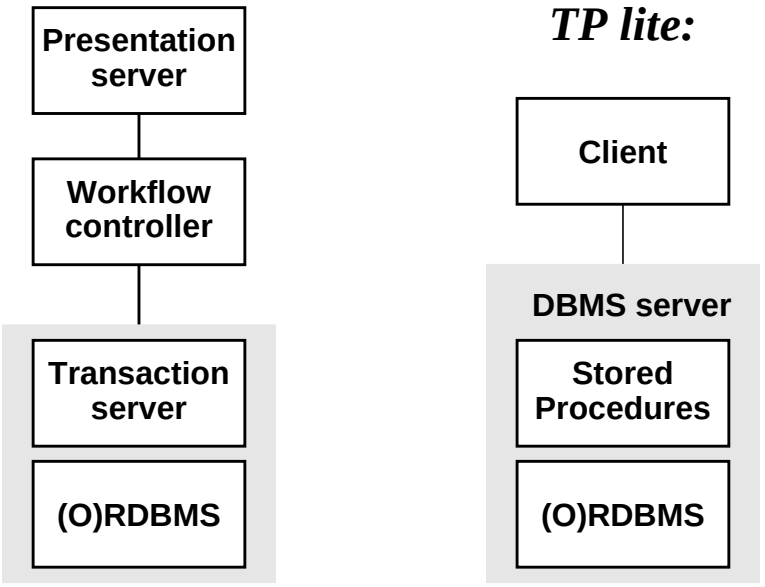
- Abbildung von Servern auf Prozesse
  - Natürliche Aufteilung (3-Tier-Architektur)
    - separate Prozesse für Presentation-Server (auf Workstation, PC oder LAN-Server)
    - separate Prozesse für Workflow-Controller, die mit den Presentation-Servern über Nachrichten kommunizieren und auf LAN-Server oder Back-End-Systemen laufen
    - Verteilung von Transaction-Server auf Prozesse auf Back-End-Systemen, 'nahe' den Daten; sie kommunizieren mit den Workflow-Controllern und miteinander über Nachrichten
  - Vorteile dieser (von den meisten modernen TP-Monitoren genutzten) Aufteilung
    - flexible Verteilung einzelner Funktionen/Prozesse auf Rechnerknoten
    - flexible Konfiguration (Lokation hinsichtlich der Optimierung von Performance, Verfügbarkeit, einfacher Administrierbarkeit, etc.)
    - flexible, unabhängige Kontrolle jeder der drei Ebenen
    - Schutz der Speicherbereiche durch Nutzung verschiedener Prozesse
  - Nachteil
    - Senden einer Nachricht: 1500-15000 Instruktionen
    - Prozeduraufruf: ~50 Instruktionen
    - beachte: jeder Request erfordert die Einbeziehung aller drei Ebenen

Betriebssystem-Prozesse (4)

- Server-Klassen
  - Menge von (single-threaded) Prozessen, die alle das selbe Programm ausführen statt Multi-Threading
  - Vorteile
    - keine Blockierung bei synchronen I/O-Operationen möglich
    - keine Sicherheitsprobleme durch gemeinsame Speicherbereiche
    - das Fehlschlagen eines einzelnen Prozesses beeinflusst die anderen Prozesse der Server-Klasse nicht
  - erfordert Dispatcher
    - Ziel: Lastbalancierung
    - Möglichkeit:
      - zufällige Zuordnung
      - Nutzung einer Eingabe-Queue für die gesamte Klasse
      - spezieller Routing-Server als Repräsentant der Server-Klasse (bedeutet aber zusätzlichen Kontext-Switch)
  - werden nur für Transaction-Server genutzt
    - ACMS (Digital) und Pathway/TS (Tandem) nutzen Server-Klassen für Transaction-Server und Multithreading im Workflow-Controller

DB-Server vs. TP-Monitore

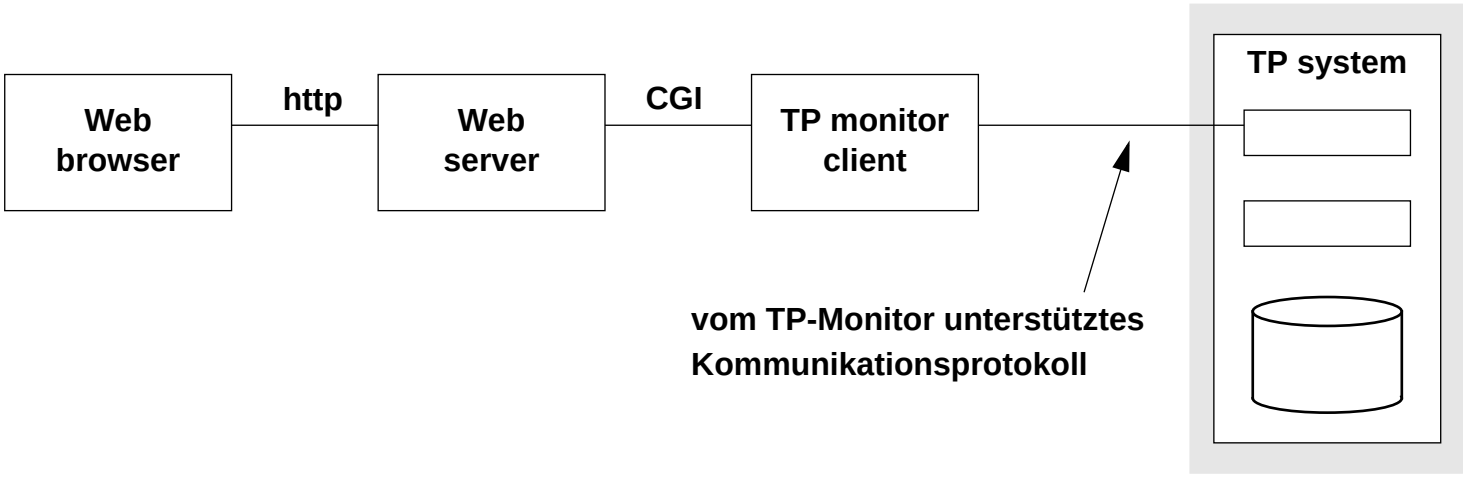
- TP lite



  - Probleme von TP-lite
    - Skalierbarkeit und Interoperabilität
    - Einschränkungen bzgl. Sprachen für DBS-Erweiterungen, z. B.
      - Aufrufe weiterer Prozeduren/Systeme
      - Datentyp-Einschränkungen
    - eingeschränkte Unterstützung für Entwicklung und Debugging
    - oft eingeschränkte Möglichkeiten bzgl. verteilter Transaktionen (z. B. nur auf dasselbe DBMS)
    - bzgl. des speziellen Produkts optimierte aber eingeschränkte Kommunikationsmöglichkeiten
  - aufgrund der guten Performanz geeignet für kleinere Anwendungen, die weniger Skalierbarkeit und Interoperabilität fordern

Transaktionen im Internet

- Kopplung von Web-Server und TP-Monitor-Funktionalität



Zusammenfassung (1)

- TP-Monitor: Entwicklung, Deployment und Administration von TP-Anwendungen
- Komponenten
  - Presentation Server
    - Interaktion mit dem Endbenutzer über Menus und Masken
      - nimmt eigentlichen Request entgegen
      - konstruiert Nachricht für Workflow-Controller
      - verantwortlich für Authentifizierung (und ggf. geographic entitlement)
  - Workflow Controller
    - Verbesserung der Skalierbarkeit
    - Routing
    - dekodiert Request-Nachricht
    - lokalisiert Transaktionsprogramm(e) und ruft diese auf
    - Directory-Service hilft, die symbolischen Request-Namen auf Server-IDs abzubilden
    - optional kann Parameter-based Routing zur Abbildung genutzt werden
    - fasst ggf. die Aufrufe mehrerer Transaction-Server zu einer Transaktion zusammen (Optionen: chained, unchained)
    - Exception-Handling bei Transaktions- und Systemfehlern
  - Transaction Server
    - führen die eigentlichen Requests auf den Daten aus

Zusammenfassung (2)

- Abbildung der ’3 Teile des TP-Monitors’ auf Prozesse
  - Trade-off: Inter-Prozess-Overhead vs. Flexibilität der Verteilung und Konfigierung der Komponenten
- Multi-Threading
  - BS-Threads
    - effektivere Nutzung von Mehrprozessorsystemen
    - TP-Monitor muss nicht synchrone I/Os abfangen
  - TP-Monitor-Threads
    - vermeidet Aufwand (Instruktionen für Thread-Switches)
  - Alternative: Server-Klassen
- TP-lite als Konkurrenz von TP-Monitoren
  - jedoch TP-Monitore überlegen in
    - Workflow-Kontrolle
    - Server-Klassen
    - Verteilung von Transaktion auf verschiedene RM-Typen
    - Verwaltungs- und Administrationsfunktionen
- mittlerweile große Bedeutung von TP-Monitor-Funktionalität für Web-Anwendungen