2. Vom System zum Modell

Dieser Teil der Vorlesung behandelt

- eine kurze Einführung in die behandelten Systeme und deren Komponenten (verteilte Systeme, Protokolle, Client-Server Systeme)
- die Modellierung des Benutzerverhaltens und der Interaktion zwischen System und Benutzer
- den Begriff der Leistung und Definition von Leistungsanforderungen
- erste Ansätze und Formeln zur Berechnung einfacher Leistungsmaße
- die Beschreibung eines Vorgehensmodells bei der Kapazitätsplanung

sicherlich keine vollständige Einführung in
zu analysierenden Systeme
(dafür gibt es andere Vorlesungen)

aber eine Erklärung der Basismechanismen,
um Leistungsmodelle zu formulieren

2.1 Struktur des Umfelds

Computer- und Kommunikationsysteme
wachsen immer stärker zusammen

kaum vorstellbare Dynamik in der Steigerung von
Rechenkapazität und Kommunikationsbandbreite
bringt immer mehr Rechenkapazität zum Benutzer

Moore's Gesetz:

Geschwindigkeit von Mikroprozessoren verdoppelt sich alle 18 Monate

wird durch Vergangenheit bestätigt

Projektion in die Zukunft:

- Computer im Jahr 2050 sind 11 Milliarden mal schneller als heute
- konservative Schätzungen gehen immer noch vom Faktor 100000 in 50 Jahren aus

Weiterhin kann man davon ausgehen,
dass die Vernetzung von Rechnern (aller Art)
immer weiter fortschreitet

Client-Server Modell entstand aus Verknüpfung Computer- und Kommunikationssysteme und bilden die Basis der meisten kommerziellen Anwendungen

2.1.1 Netze und Protokolle

Ende 60er Jahre Beginn der Entwicklung mit ARPANET

Begriff Internet entstand 1983 bei Teilung des ARPANETs in militärischen und zivilen Teil

Internet ist Menge von wide area networks (WANs) mit den Basisprotokollen TCP/IP

Netzwerktypen

man unterscheidet nach Ausdehnung

- wide area networks (WANs)
- (teilweise) metropolitan area networks (MANs)
- local area networks (LANs)

WANs

- Ausdehnung einige Kilometern bis ganzer Kontinent
- Basistechnik paket switching (Paketvermittlung)
- Nachricht wird vom Sender in Pakete unterteilt
- Pakete haben eine maximale Größe
- Pakete enthalten Adressinformation, Sequenznummern im *Header*
- Empfänger setzt Pakete zur Nachricht zusammen
- auf dem Weg vom Sender zum Empfänger durchlaufen Pakete Zwischenknoten
- unterschiedliche Technologien für Paketvermittlung existieren (X25, ISDN, ATM etc.)

LANs

Typische Ausdehnung: ein Gebäude

Typische LAN Technologien

- 10 Mbps - 1 Gbps Ethernet
- 16 Mbps Token Ring
- 100 Mbps Fiber Distributed Data Interconnect (FDDI)

Ethernet

- Bustopologie (Koaxialkabel)
- alle Teilnehmer können alle Pakete empfangen
- keine zentrale Kontrolle
- Zugangsprotokoll Carrier Sense Multiple Access / Collision Detection (CSMA/CD)
- Sender hört Kanal ab
  - ▷ bei Sendewunsch und belegtem Kanal wird gewartet
  - ▷ bei freiem Kanal wird gesendet
- durch endliche Ausbreitungsgeschwindigkeit kann es zu Kollisionen kommen
  - ▷ Sender entdecken Kollision unterbrechen Übertragung
  - ▷ warten zufälliges Zeitintervall (welches mit der Zahl der Übertragungsversuche wächst)
  - ▷ versuchen neue Sendung

Verhalten von Ethernet

sehr geringe Verzögerung bei geringer Last

bei steigender Last steigende Kollisionwahrscheinlichkeit,
wachsende Verzögerungszeiten, fallender Durchsatz

$\Rightarrow$ CSMA/CD kann keine maximale Verzögerung
garantieren!

Token Ring

- Ringtopologie
- Sender schickt Paket in den Ring
- Empfänger kopiert Paket
- Sender nimmt Paket vom Ring
(und testet auf Übertragungsfehler)
- um Kollisionen zu vermeiden rotiert Sendeberechtigung
(Token) im Ring
- Token wird zyklisch von Station zu Station gegeben

FDDI

- Lichtwellenleitungen
- zwei Ringe mit unterschiedlichem Datenfluss
($\Rightarrow$ Fehlertoleranz)

Verhalten von Token Ringen

Stationszahl bestimmt Verzögerung

maximale Verzögerung kann garantiert werden

Limitierende Faktoren in LANs

physikalisch Leitungslänge

Leistungsbeschränkung durch Anzahl Stationen

$\Rightarrow$ Unterteilung in Segmente

Struktur des Gesamtsystems

Verbindung von LANs über

- Router oder Brücken
- backbone Netze

(i.a. FDDI oder 100 Mbps - 1 Gbps Ethernet)

oft geschaltete Verbindungen

Verbindung zum WAN über Standleitungen

- T1 1.544 Mbps
- T3 45 Mbps

Dimensionierung der Leitungen ein Kapazitätsplanungsproblem!

Verbindungen einzelner Rechner WAN

- Modem 14.4 bis 56 Kbps
- ISDN 128 Kbps
- High Bit Rate Digital Subscriber (HDSL) 1.544 Mbps
- asymmetrischer Version (ADSL) 640 Kbps senden und 8 Mbps empfangen
- TV Kabel 10 Mbps

Drahtlose LANs

Verstärkter Zugriff auf Netze über Funknetze

Standardzugriffsprotokoll: IEEE802.11

Medienzugriffsverfahren CSMA/CA
(Carrier Sense Multiple Access/Collision Avoidance)

Zugriffsverfahren ähnlich zum Ethernet aber

explizites Senden von Bestätigungen

keine Kollisionserkennung (wegen "hidden terminals")

Ablauf einer Übertragung: Sender tastet Medium ab

- bei freiem Kanal
  - direktes Senden der Nachricht mit Angabe der Header oder
  - Senden einer kurzen RTS (request to send) Nachricht zur Reservierung und Bestätigung dieser Nachricht mittels CTS (clear to send) Nachricht vom Empfänger
- bei belegtem Kanal
  - Verzögerung mindestens um bekannte Kanalbelegungszeit plus zufällige Backoff-Zeit
  - bei Kollisionen auf dem Netz Verdopplung der mittleren Backoff-Zeit
- beim Auftreten von Kollisionen
  - durch ausbleibende Bestätigungen erfolgt Neuübertragung

Leistungsverhalten IEEE802.11 ähnlich Ethernet-Protokoll

Protokolle

Kommunikation zwischen Prozessen über Computernetze erfolgt nach einer Menge von Regeln, dem *Protokoll*

Aufgaben eines Protokolls

- Festlegung der Adressierung und Wegewahl (routing)
- Fehlererkennung und Fehlerbeseitigung
- Sequenzkontrolle eintreffender Nachrichten
- Flusskontrolle zur Regelung der Übertragungsrate

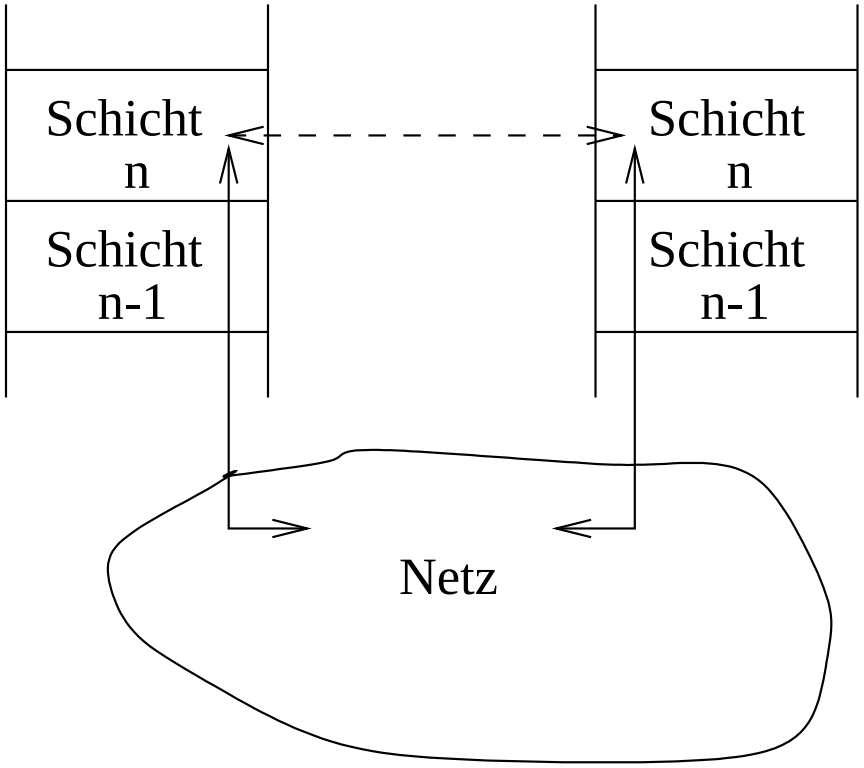
Man unterscheidet

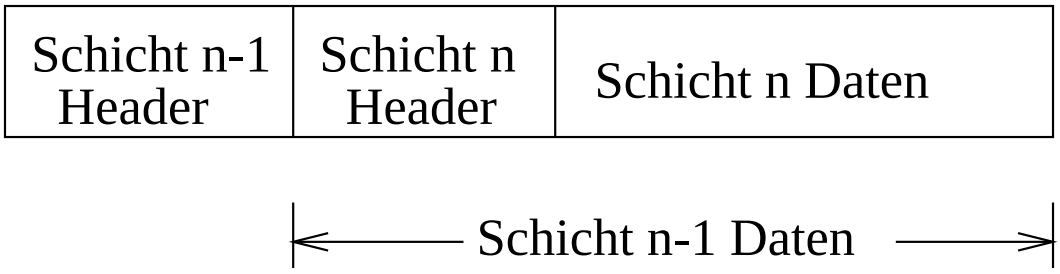
- verbindungslose Übertragung
  - ▷ Nachrichten werden unabhängig durch das Netz geleitet
  - ▷ Nachrichten können in anderer Reihenfolge beim Empfänger eintreffen
- verbindungsorientierte Übertragung
  - ▷ erst virtuelle Verbindung aufbauen
  - ▷ dann Übertragung über die Verbindung

Protokolle haben

- *Syntax*, d.h. Regeln welche Nachrichten gesendet/ empfangen werden, die Nachrichtenformate etc.
- *Semantik*, d.h. Aktionen die bei Eintreten von Ereignissen erfolgen müssen

Struktur von Kommunikationsprotokollen





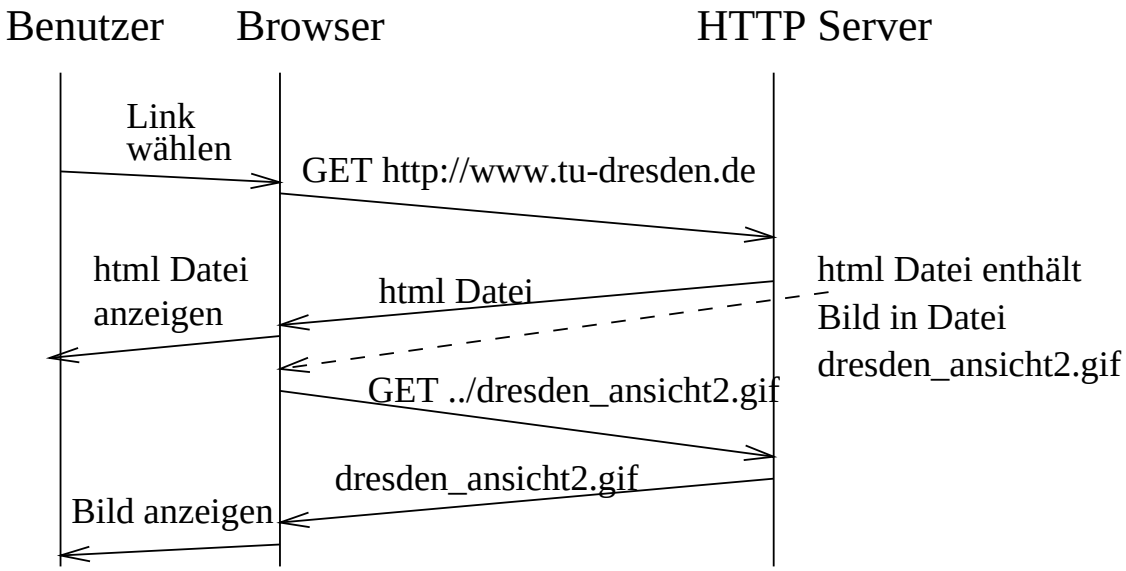
7 Schichtenmodell der ISO

Kommunikation in Schicht n

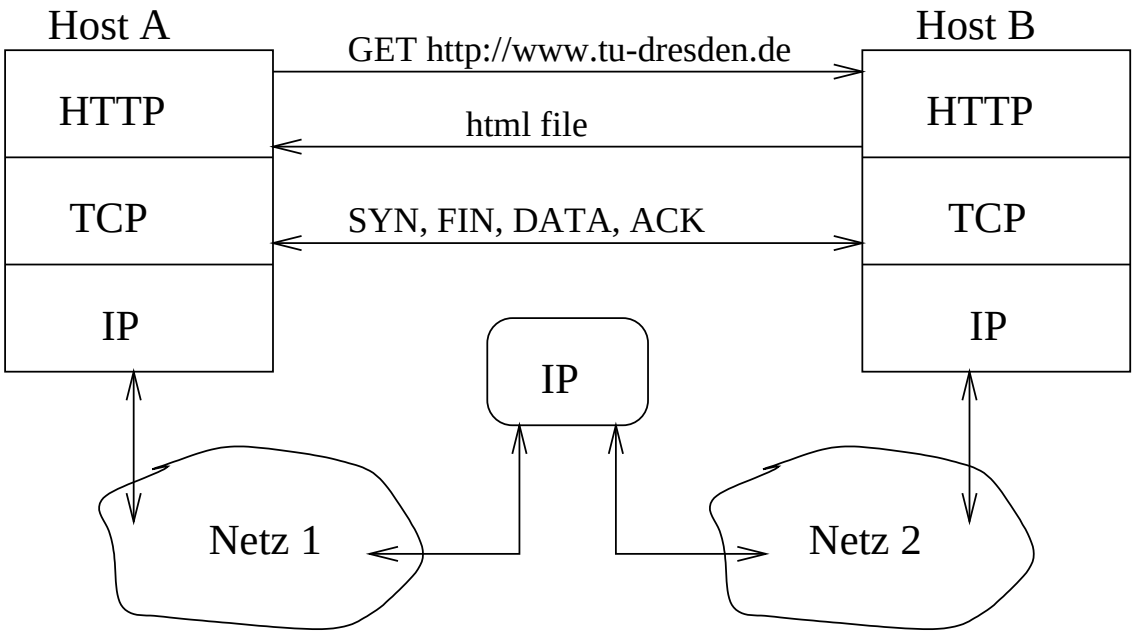
- mit Kommunikationspartner aus Schicht n
- durch Benutzung von Diensten aus Schicht $n-1$

Kommunikation im Web

Beispiel HTTP Transaktion



Ablauf im Netz

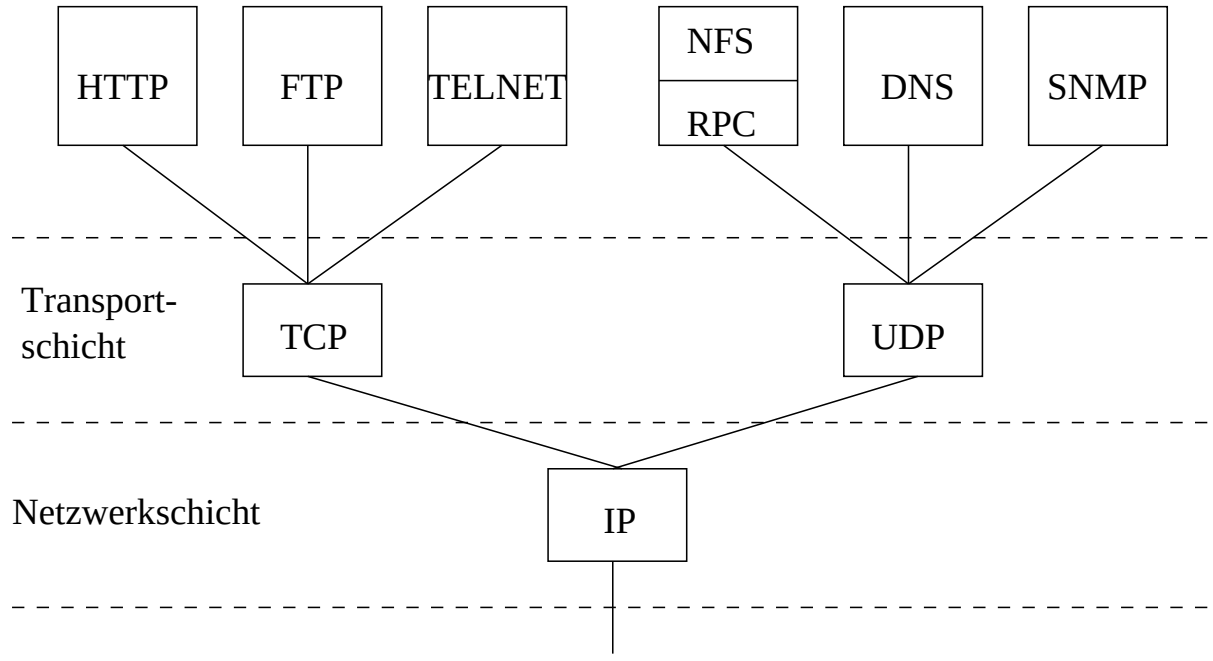


- **HTTP** (Hypertext Transfer Protocol) ist ein Anwendungsprotokoll, welches das Internet Transport Protokoll TCP benutzt

- **TCP** (Transmission Control Protocol) ist ein verbindungsorientiertes Transportprotokoll, welches Kommunikation zwischen zwei Rechnern im Internet realisiert und Pakete über IP überträgt

- **IP** (Internet Protocol) ist Netzwerkprotokoll, welches die Paketformate beschreibt und die Übertragung der Pakete durch das Netz realisiert

Übersicht über Internet Protokolle



IP Protokoll

jeder Rechner im Internet hat eindeutige Adresse
(IP-Adresse)

32-Bit Nummer der Form 141.76.82.163

(jede Zahl variiert zwischen 0 und 255 und
beschreibt 8 Bit)

- erster Teil (Präfix) beschreibt Netznummer
- zweiter Teil (Suffix) beschreibt den Rechner

Daten werden als IP-Datagramme bezeichnet

IP garantiert keine

- verlustfreie Übertragung und
- auch keine Reihenfolge

IP Header umfasst 20 Bytes
(davon 8 Bytes für Quell- und Zieladresse)

wichtige Aufgabe Routing von Quelle zum Ziel

$\Rightarrow$ Rechner im Internet müssen Routingtabelle zur
Weiterleitung von Paketen speichern

Heutige Version IPv4 erreicht Grenzen
der 32 Bit Adressierung

Neue Version IPv6 mit 128 Bit Adressierung und
Unterstützung für Audio- und Video-Übertragung
durch Priorisierung von Paketen

TCP-Protokoll

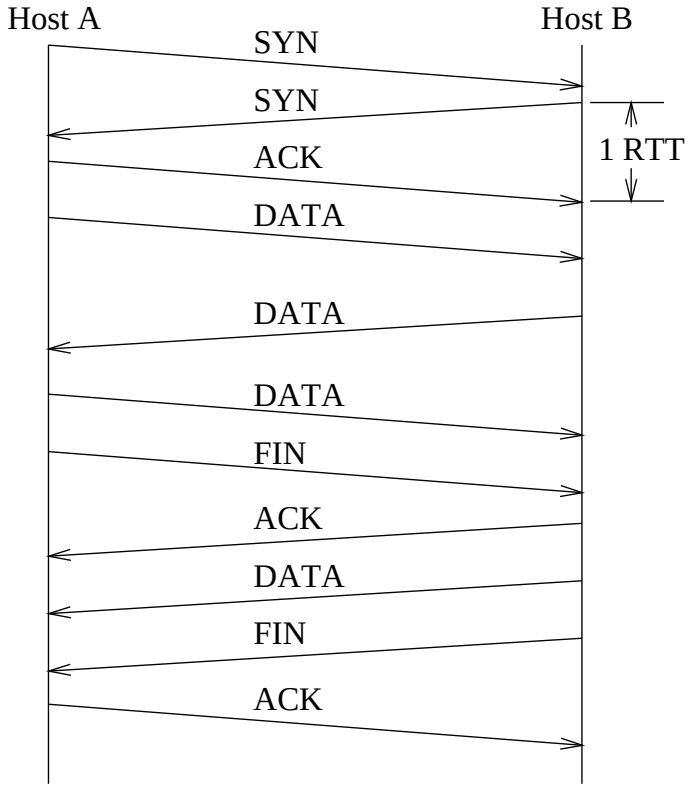
Bereitstellung einer

- verbindungsorientierten
- zuverlässigen
- flusskontrollierten
- vollduplex

Ende zu Ende Verbindung

PDUs werden Segment genannt, Header 20 Bytes lang

3-Wege Handshake zum Verbindungsmanagement



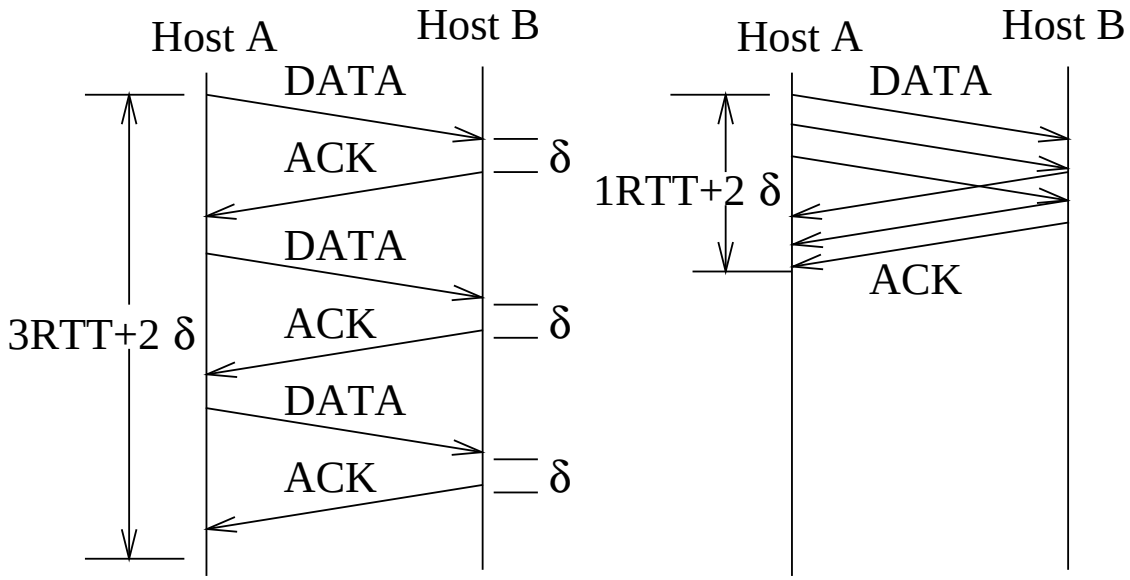
RTT (round-trip time)

TCP Fehlerkontrolle über

- Acknowledgements
- Timeouts
- Übertragungswiederholung

Flusskontrolle durch

- Fenstermechanismus (maximale Anzahl sendbarer Pakete bevor eine Bestätigung eintreffen muss)



In den meisten Netzen gilt $RTT \gg \delta$

Durchsatz X_1 für Fenstergröße 1:

$$X_1 = 3/(3 \cdot RTT + 2 \cdot \delta) \approx 1/RTT$$

Durchsatz X_3 für Fenstergröße 3:

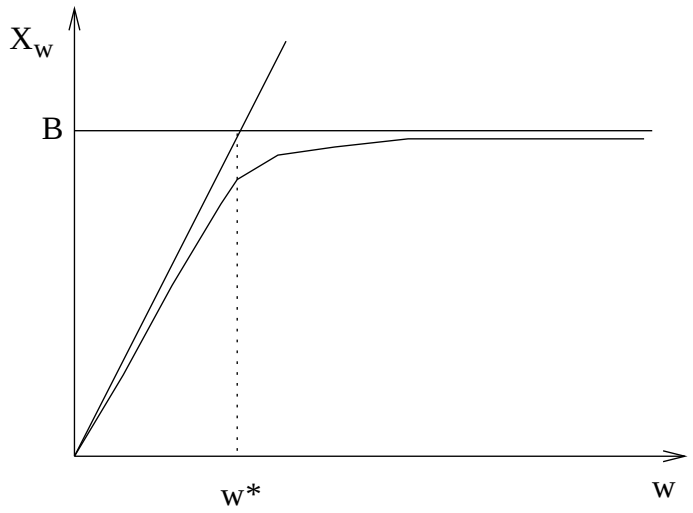
$$X_3 = 3/(RTT + 2 \cdot \delta) \approx 3/RTT = 3 \cdot X_1$$

Bandbreite limitiert den maximalen Durchsatz

$$X_w = \min(B, w \cdot X_1)$$

mit B Bandbreite der Verbindung

Verlauf des Durchsatzes



Optimale Fenstergröße $w^* = \lceil B/X_1 \rceil$

TCP benutzt adaptiven Mechanismus

- Start mit kleiner Fenstergröße
- Vergrößerung in Abhängigkeit von gemessenen RTT

Netztyp	Bandb. Byps	RTT Sek	X_1 Byps	X_{w^*} Byps	w^*	X_{w^*}/X_1
Ethernet	1090000	0.0007	1090000	1090000	1	1.000
Fast Ethern.	12500000	0.0007	2085714	12500000	6	5.993
Slow Intern.	12750	0.1610	9068	12750	2	1.406
Fast Intern.	127500	0.0890	16404	127500	8	7.772

2.1.2 Client-Server Systeme

Aufteilung von Arbeit auf Clients und Server

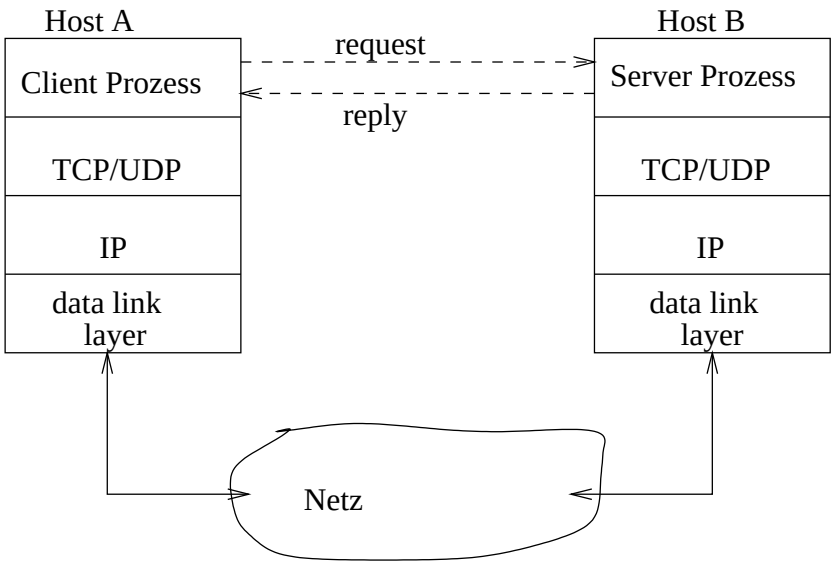
Client-Prozesse:

- laufen lokal auf einer Workstation mit GUI
- rufen Services auf, die von einem oder mehreren Server bereitgestellt werden
- führen Teile des Codes der Anwendung aus

Server-Prozesse:

- laufen üblicherweise auf einer größeren Maschine
- führen eine Menge zusammengehöriger Services aus
- warten auf Anfragen von Client-Prozessen, führen diese aus und versenden die Antworten
- initiieren keine Kommunikation mit Client-Prozessen

zwischen Client und Server *request-reply Protokoll*

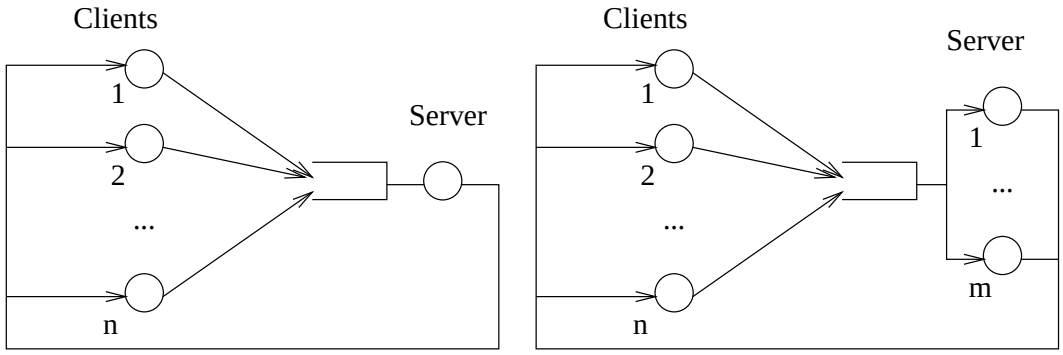


Server empfangen Anfragen von vielen Clients

- bei rein sequentieller Bedienung
  - ▷ geringe Auslastung des Server-Rechners
  - ▷ geringer Durchsatz
  - ▷ linear mit der Last steigende Antwortzeiten

Verwendung von mehreren Threads für die Ausführung der Anfragen und ein Prozess/Thread für die Einordnung neuer Prozesse in die Warteschlange

Modellvorstellung



Threads laufen quasi-parallel (Konkurrenz um Ressourcen)

Durchsatzgrenze durch 100% Auslastung einer Ressource

Beispiel:

Ankunftsrate 2.5 Anfr./Sek. 0.07 Sek. CPU-Zeit und
0.2 Sek. Plattenzugriffs-Zeit

Threads	1	2	3	4	5
Antwortzeit	0.831	0.526	0.492	0.486	0.485

Typen von Servern

- File Server
- Datenbank Server
- Anwendungs Server
- Software Server
- Web Server
- ...

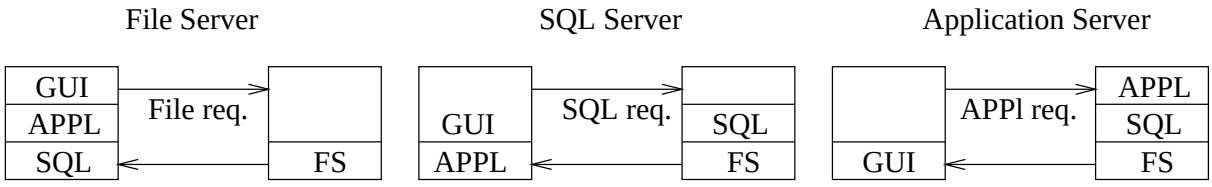
weit verbreiteter File-Server Network File System (NFS)

- erlaubt Zugriff auf Dateien unterschiedlicher Netzwerke
- basiert auf TCP oder UDP

Datenbankserver erlauben Zugriff auf eine oder mehrere Datenbanken über SQL Anfragen

Anwendungs Server stellen Prozeduren zur Verfügung, auf die mittels remote procedure call (RPC) zugegriffen werden kann

Möglichkeiten der Verlagerung von Arbeit vom Client zum Server



Komplexität der Anfragen nimmt zu, aber Anzahl der Anfragen nimmt ab

Architekturentscheidungen

Zweistufige- oder dreistufige Architektur

Typische zweistufige Architektur

- GUI und Anwendung auf dem Client
- SQL Server auf dem Server

Typische dreistufige Architektur

- GUI auf dem Client
- Anwendung auf dem Anwendungs Server, welcher Client für einen
- SQL Server ist

Entscheidung über *fat* oder *thin* Clients (Netzwerkcomputer versus Workstation)

Instrument zur Leistungssteigerung

Caches auf unterschiedlichen Ebenen

- Caches speichern Server Daten
- sind schneller vom Client aus zugreifbar (oder liegen direkt auf dem Client, z.B. Caches in Browsern)
- Caching kann Leistung deutlich steigern in Abhängigkeit von Trefferwahrscheinlichkeit (*hit ratio*) (= Wahrscheinlichkeit, dass ein angefragtes Datum im Cache liegt)
- Preis dafür: zus. Aufwand zur Konsistenzerhaltung, insbesondere bei Datenänderung durch Client

2.2 Modellierung des Benutzerverhaltens

Im wesentlichen Modellierung des Verhaltens von Benutzern in web-basierten Systemen, Methodik aber auch auf allgemeinere Fälle anwendbar (z.B. Software-Agenten)

Ziel der Benutzermodellierung:

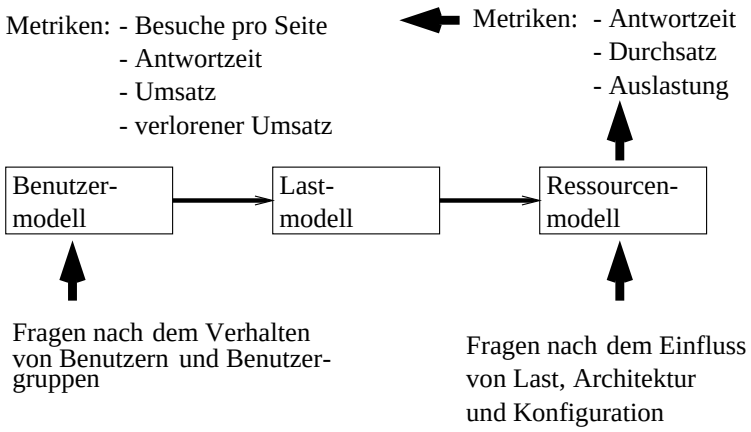
Nachbildung des typischen Verhaltens eines Benutzers in einem Client-Server System

Verhalten wird charakterisiert durch aufeinander folgende Aufrufe von Benutzerfunktionen

Beschreibung des Benutzerverhaltens durch

Benutzerverhaltensgraph (*BVG*)

BVG beschreibt detailliert das Benutzerverhalten:
für die hier verwendeten Analysemethoden reicht aber eine aggregierte Sicht, die durch das *Benutzerbesuchsmodell* (*BBM*) beschrieben wird



Generierung eines BVGs am Beispiel einer Online Buchhandlung:

Verfübare Funktionen:

Home Startseite der Buchhandlung

Search suchen eines Buches

Browse blättern im Katalog

Select auswählen eines Buches

Login "betreten" der Buchhandlung

Register Registrierung als Kunde

Add Kauf eines Buches

Pay bezahlen der Bestellung

Exit implizit vorhandener Zustand, der das Ende einer Transaktion bezeichnet

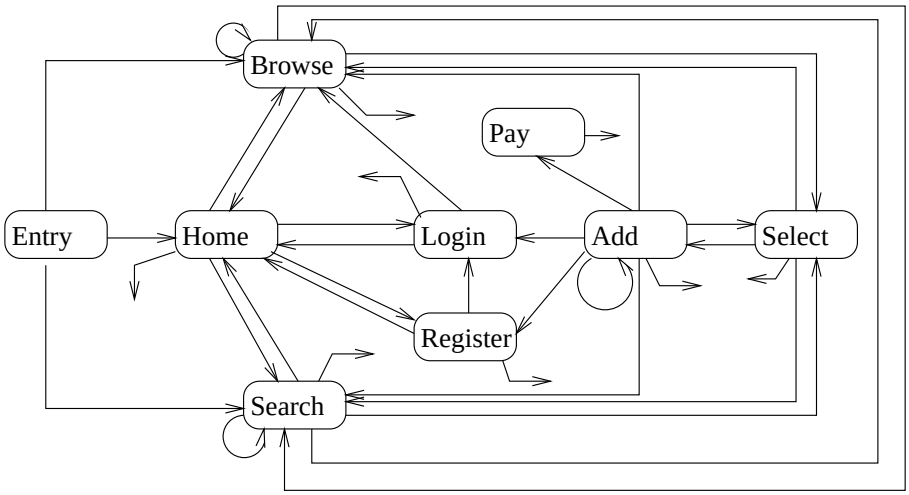
Entry virtueller Anfangszustand

Zustände sind im BVG über gerichtete Kanten verbunden

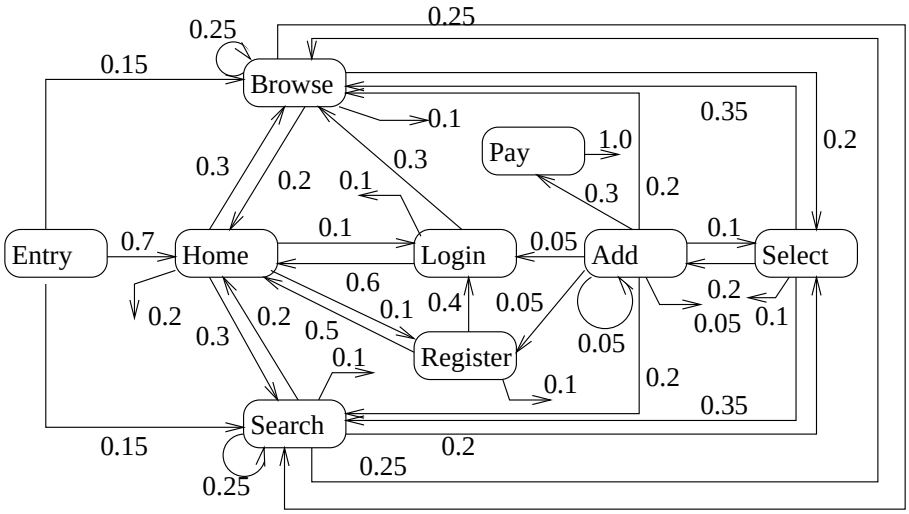
Semantik der Kanten: Nachdem der Kunde eine Aktion am Anfang der Kante durchgeführt hat wird (möglicherwiese) die Aktion am Ende der Kante durchführen

- vom Zustand Entry gehen nur Kanten ab
- der Zustand Exit wird nicht gezeichnet, wird aber mit allen Kanten ohne Zielzustand verbunden

BVG des Beispiels



Graph in dieser Form enthält noch keine
quantitative Information
$\Rightarrow$ zusätzliche Verzweigungswahrscheinlichkeiten



damit kann jeder BVG durch eine substochastische $n \times n$ Matrix P beschrieben werden

Erstellung des BVG:

Generierung des statischen Teils
(ohne Verzweigungswahrscheinlichkeiten)

- Identifizierung der Basisfunktionen, die ein Kunde aufrufen kann
- Verfeinerung der angebotenen Funktionen
  bzgl. Ressourcenanforderungen
- Bestimmung der möglichen Transitionen
  z.B. durch Analyse der angebotenen Web Seiten
- Bestimmung der Verzweigungswahrscheinlichkeiten durch Messungen oder Log-Dateien u. U. Aufteilung in mehrere Benutzerklassen (mehr dazu in Kap. 4 der Vorlesung)

Aus dem BVG ableitbare Resultate:

- mittlere Anzahl Aufrufe einer Funktion pro Besucher
- Umsatz pro Besucher
- mittlere Anzahl geladener Seiten pro Besucher
- Wahrscheinlichkeit nach Funktion x irgendwann vor Verlassen des Servers Funktion y aufzurufen

wir berechnen die mittlere Anzahl von Funktionsaufrufen pro Benutzer

Idee der Benutzermodellierung lässt sich auch auf andere Anwendungen übertragen
Beispiele

- Programme:
  - Programm besteht aus Funktionsaufrufen, die jeweils Ressourcen belasten
  - jeder Knoten beschreibt eine Funktion
  - datenabhängige Verzweigungen werden durch probabilistische Verzweigungen ersetzt
  - Ableitung der Graphstruktur aus der Spezifikation und/oder dem Programmcode

- Automatisierungssysteme:
  - Module zur Steuerung und zum Monitoring
  - Module werden durch BVGs beschrieben
  - Gesamtsystembeschreibung besteht aus mehreren BVGs
  - Synchrone Kommunikation wird im Modell durch probabilistische Verzweigungen ersetzt

Struktur der Matrix P

- Zeile 1 beschreibt die aus Entry abgehenden Transitionswahrscheinlichkeiten
- P ist substochastisch, d.h. jede Zeilensumme ist ≤ 1.0 und die Matrix enthält nur positive Werte
- Zeilen und Spalten von P lassen sich nicht so umordnen, dass eine stochastische Untermatrix auf der Diagonalen entsteht

aus der Theorie der absorbierenden Markov Ketten folgen die Eigenschaften

- die Matrix $N = (I - P)^{-1} = \sum_{k=0}^{\infty} P^k$ existiert
- Element $N(i, j)$ beschreibt die mittlere Anzahl von Besuchen in Zustand j vor Verlassen des Systems, wenn der aktuelle Zustand i ist
- da jeder Kunde seinen Besuch mit dem Zustand Entry beginnt, entspricht $N(1, j)$ der mittleren Anzahl der Aufrufe von Funktion j bei einem Besuch
- $\sum_{j=2}^{n} N(1, j)$ entspricht der mittleren Anzahl an Funktionsaufrufen eines Besuchers

Für unser Beispiel ergibt sich

Funktion	Aufrufe	Funktion	Aufrufe
Entry	1.000	Login	0.269
Home	1.826	Pay	0.056
Browse	2.213	Register	0.192
Select	0.888	Add	0.187
Search	2.132		

den Vektor der mittleren Aufruf- oder Besuchhäufigkeiten bezeichnet man als Benutzerbesuchsmodell (BBM)

Praktisches Problem:

Identifizierung von Benutzern und Sitzungen

HTTP ist zustandslos!!

Identifizierung über **Cookies**

- Server sendet mit seiner Antwort eine eindeutige Identifizierung (Cookie) an den Client
- Cookies werden vom Client gespeichert und bei weiteren Anfragen mit zum Server geschickt

dadurch kann Server Client identifizieren

- Zuordnung von Sitzungen mit Hilfe eines vordefinierten Zeitfensters

$\Rightarrow$ aus diesen Daten sind gewünschten Werte ermittelbar

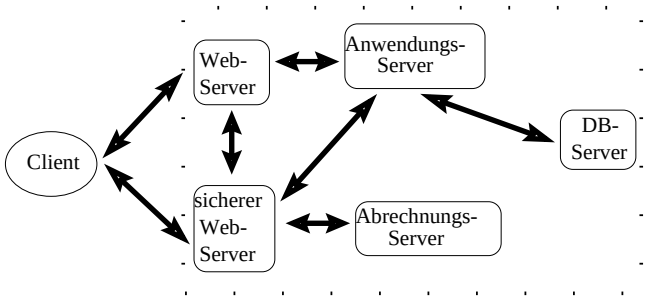
Struktur der Interaktion zwischen Client und Server

zur Zuordnung von Benutzerfunktionen zu Diensten der Ressourcen Interaktionsmodelle

- Server-orientiert: Client kommuniziert mit Server-Prozess über RPC
  - Server stellt benötigte Funktionen zur Verfügung
  - Server Stub sorgt dafür, dass Funktionsaufrufe den richtigen Server Prozessen zugeordnet werden
- Objekt-orientiert: Dienste werden von Objekten angeboten
  - Objekte können sich irgendwo im verteilten System befinden
  - Object-Request-Broker stellt Verbindung zwischen Client und "Server" her (Realisierung z.B. Corba)

Objekt-orientierte Ansatz ist flexibler fügt aber zusätzliche Interaktionen ein $\Rightarrow$ Leistungsverluste

Interaktion zwischen E-Commerce Anbieter und Kunden



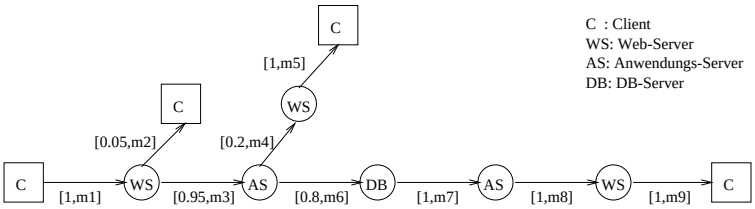
Interaktion zwischen mehreren Servern und einem Client

Beschreibung der Interaktion zwischen Client und Server über *Client-Server Interaktions Diagramme* (*CSIDs*)

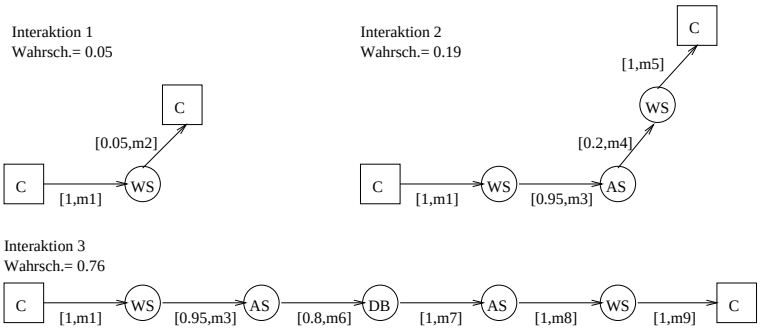
CSIDs sind gerichtete, azyklische und bewertete Graphen mit

- zwei Arten von Knoten
  - quadratische Knoten beschreiben Clients und sind jeweils am Ende jedes Pfades zu finden
  - kreisförmige Knoten beschreiben Server
- Kanten sind gerichtet und bewertet mit
  - Wahrscheinlichkeiten (die Summe der abgehenden Kantenwahrscheinlichkeiten muss 1.0 ergeben) und
  - der Größe der verschickten Nachricht in Byte

Beispiel

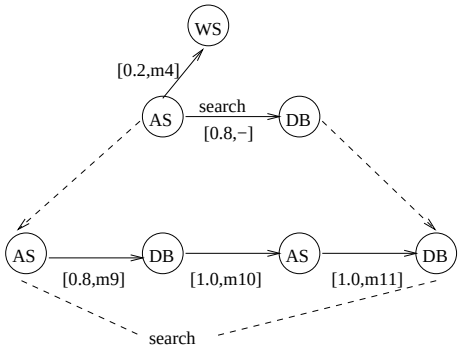


Interaktionsdiagramm beschreibt 3 mögliche Abläufe



Für komplexe Interaktionen können CSIDs hierarchisch strukturiert sein: Eine Interaktion (in diesem Fall ohne Angabe der Nachrichtengröße) beschreibt eine Sequenz von Interaktionen

Beispiel



CSIDs beschreiben im wesentlichen die Interaktion zwischen Akteuren (Client und Servern) und die dabei auftretenden Nachrichtengrößen

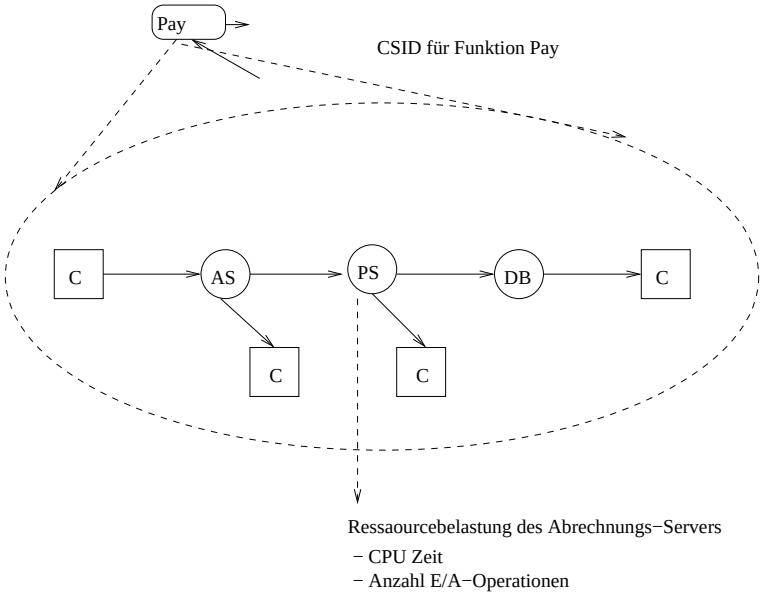
Aus einem CSIDs ableitbare Resultate

- Wahrscheinlichkeiten einzelner Abläufe
- mittlere Anzahl der verschicken Bytes pro Ablauf oder über alle möglichen Abläufe
- erste Schätzungen von Übertragungszeiten, falls verwendete Netze bekannt (gleich mehr dazu)
- erste Schätzung der Gesamtdauern, falls zus. Dauer der Funktionaufrufe auf den Servern bekannt (gleich mehr dazu)

Abbildung BVG auf CSID

1. Aufbau des BVGs
2. Auflistung aller vom Benutzer aufgerufenen Funktionen
3. Bestimmung der benutzten Server für jede Funktion
4. Bestimmung des Nachrichtenflusses pro Funktion
5. Schätzung der Nachrichtengrößen
6. Bestimmung der Ressourcenbelastung pro Server-Knoten

Beispiel



PS Abrechnungs−Server
AS Anwendungs−Server
DB DB−Server

2.3 Leistungsanforderungen an Client-Server Systeme

Client-Server Systeme belegen verschiedene Ressourcen

- Workstations, PCs
  - ▷ Platten
  - ▷ Speicher
  - ▷ Prozessoren
- Netzwerke
  - ▷ LANs
  - ▷ WANs
  - ▷ Router

und bestehen aus zahlreichen Softwarekomponenten

- Anwendungsprogramme
- DB Manager
- BS Komponenten
- Protokolle
- GUI

alle diese Komponenten belegen Ressourcen und benötigen Zeit

Ziel der Leistungsanalyse und Grundlage für Kapazitätsplanung sind

- Auslastungen von Ressourcen
- Zeitverbräuche

Ziele diese Abschnitts:

- Einführung von Verzögerungsdiagrammen zur Ermittlung von Verzögerungszeiten in Kommunikationssystemen (2.3.1)
- Ermittlung von Bedienzeiten für Platten, Plattenarrays und Netzwerke (2.3.2)
- Einführung erster Grundlagen über Warteschlangen (2.3.3)
- Definition grundlegender Gesetze der operationalen Analyse (2.3.4)
- Vorstellung von Metriken und Leistungsmaßen für Client Server Systeme (2.3.5)
- Beschreibung zahlreicher Beispiele zur Veranschaulichung

2.3.1 Verzögerungszeiten in Kommunikationssystemen

Verzögerungsdiagramme zur Visualisierung der Verzögerungszeiten in einem C/S-System

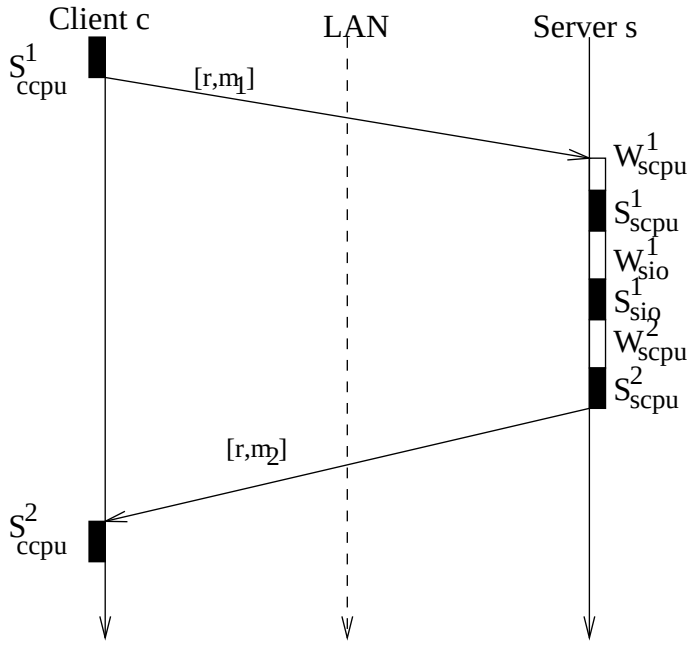
Zeitachsen: vertikale Linien auf denen Zeit von oben nach unten fortschreitet

Man unterscheidet

Kommunikationszeitachse (gestrichelte Linie) beschreibt Verzögerungen im LAN oder WAN

Verarbeitungszeitachse (durchgezogene Linie) beschreibt Verzögerungen durch Wartezeiten oder Verarbeitungszeiten beim Client oder Server

Beispiel



Nachrichten werden in der Form $[id, m]$ beschrieben, wobei id die Identifikation von Anforderung /Antwort ist und m die Größe der Nachricht (inkl. Overhead) in Bytes angibt

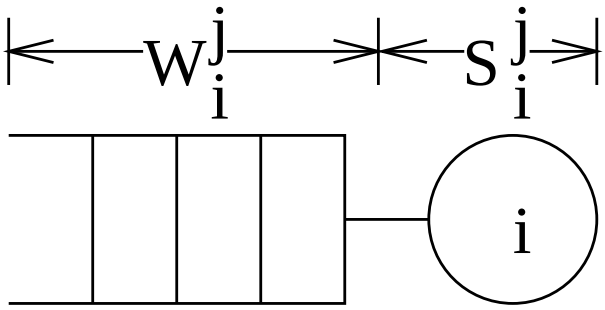
Im Beispiel kommen eine Reihe von Ressourcen vor

- LAN
- CPU Client
- CPU Server
- E/A Server

Ressourcen werden von Prozessen für ihre *Bedienzeiten* belegt

Da mehrere Prozesse auf die Ressourcen zugreifen, entstehen *Wartezeiten*

Abstrakte Sicht einer Ressource



Ein oder mehrere Bediener und ein Warteraum

Notationen

- S_i^j Bedienzeit an Ressource i beim j-ten Besuch
- W_i^j Wartezeit an Ressource i beim j-ten Besuch
- D_i Bedienanforderung an Ressource i (Summe der S_i^j)
- Q_i Gesamtwartezeit an Ressource i (Summe der W_i^j)
- R'_i Verweilzeit an Ressource i $(D_i + Q_i)$

Im Beispiel: Verweilzeit an der Server CPU:

$$R'_{scpu} = W^1_{scpu} + S^1_{scpu} + W^2_{scpu} + S^2_{scpu}$$

Für die Antwortzeit der Anforderung r gilt:

$$R_r = R'_{ccpu} + R'_{scpu} + R'_{sio} + R_{LAN}$$

Berechnung der Bedienzeiten im LAN

$$D_{LAN} = 8 \cdot (m_1 + m_2)/B$$

wobei B die Bandbreite in Bps ist

Zusammenfassung der bisherigen Formeln

$$\boxed{\begin{array}{lcl} D_i & = & \sum_{Besuche\ j} S_i^j \\ Q_i & = & \sum_{Besuche\ j} W_i^j \\ R'_i & = & D_i + Q_i \\ R_r & = & \sum_{Ressourcen\ i} R'_i \end{array}}$$

Ein Beispiel

Transaktion t in einem C/S-System

Bedienbedarf:

- 5 msek Client CPU
- 10 msek Server CPU
- 10 Blöcke a 2048 Byte von der Server Platte lesen

Damit gilt

$$D_{ccpu} = 0.005 \text{ Sek und } D_{scpu} = 0.010 \text{ Sek.}$$

Bestimmung der Bedienzeit an der Platte

- mittlere Suchzeit 9 msek
- mittlere Latenzzeit 4.17 msek
- Transferrate 20 MB/sek

Bedienzeit pro Block

$$\begin{aligned} S_d &= Suchzeit + Latenzzeit + Transferzeit \\ &= 0.009 + 0.00417 + 2048/20\,000\,000 \\ &= 0.0133\ sek \end{aligned}$$

Damit auch

$$D_d = 10 \cdot S_d = 0.133 \text{ sek}$$

(gleich mehr zur Berechnung von Bedienzeiten für Platten)

Client und Server seien mit 10 Mbps Ethernet verbunden

Nachricht vom Client zum Server umfasst ein Paket der Länge 1518 Bytes

Antwort besteht aus 7 Paketen der Länge 1518 Bytes

$$\begin{aligned} D_{LAN} &= 8 \cdot (m_1 + m_2)/B \\ &= 8 \cdot (1518 + 10626)/10000000 = 0.0097\, sek \end{aligned}$$

(gleich mehr zur Berechnung von Bedienzeiten für Netzwerken)

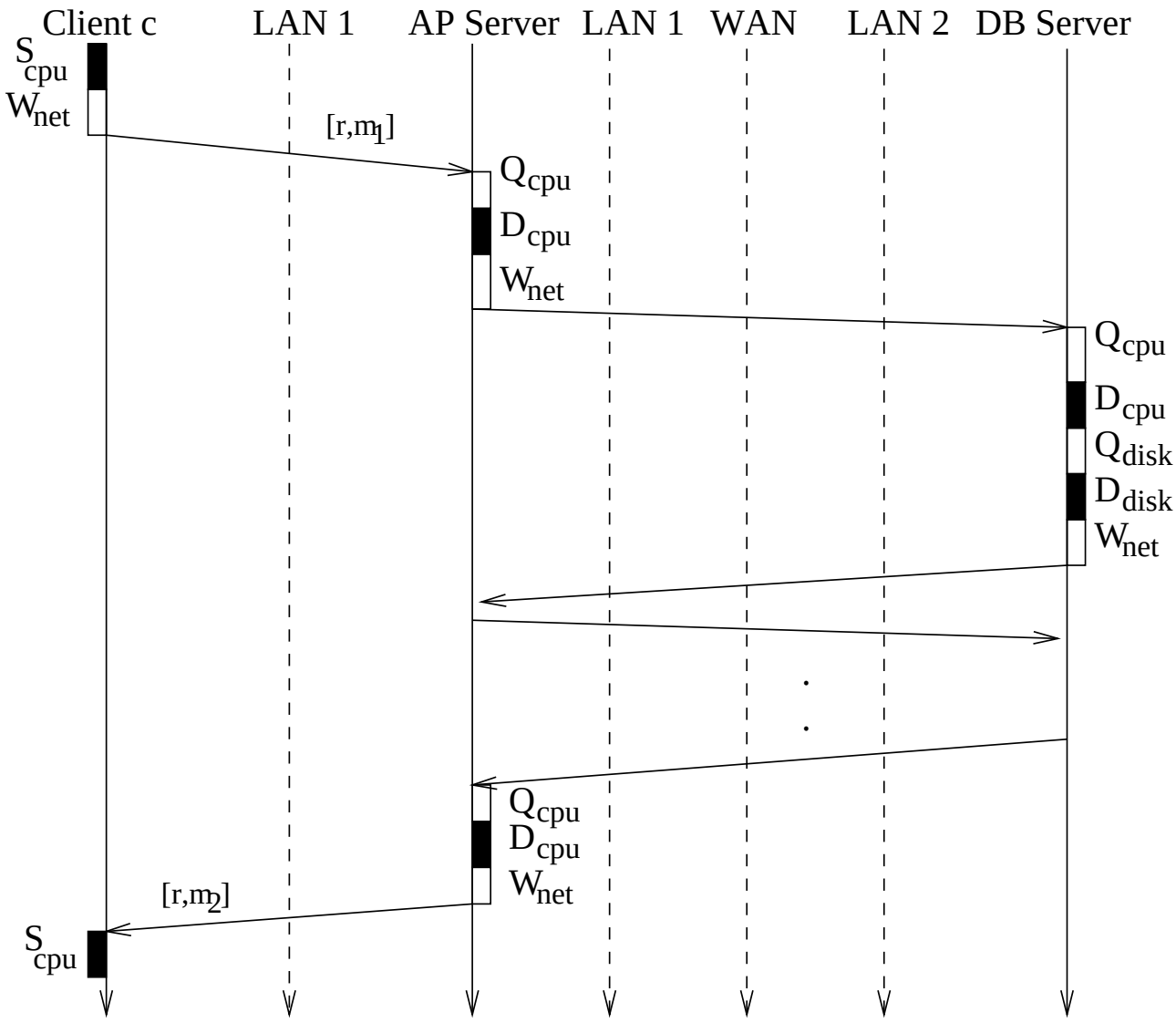
Damit gilt

$$\begin{aligned} R_t &\geq D_{ccpu} + D_{scpu} + D_d + D_{LAN} \\ &= 0.005 + 0.010 + 0.133 + 0.0097 = 0.158\, sek \end{aligned}$$

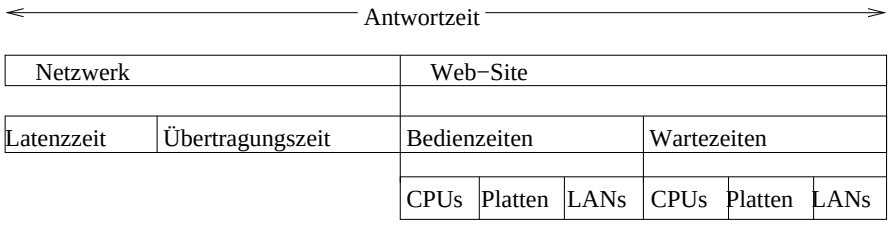
Beachte:

- Wartezeiten wurden bisher vollständig vernachlässigt
- damit sind nur untere Schranken berechenbar
- je nach Belastung können wahre Werte deutlich größer sein
- im Rahmen der Vorlesung werden wir Verfahren kennenlernen, um Wartezeiten zu bestimmen

Dreistufige Architekturen



Gesamtdarstellung der Antwortzeit



Eine Anfrage kann mehr als eine Web-Site besuchen und kann dabei mehrfach auf Ressourcen zugreifen

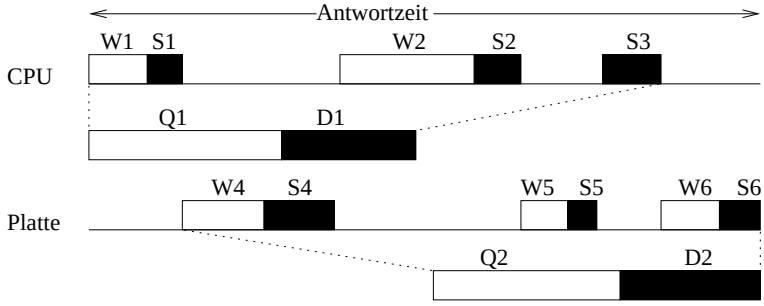
Hier Berechnung nur über Mittelwerte (Annahmen):

- Anfrage besucht Ressource i im Mittel V_i-mal
- jeder Besuch erfordert eine mittelere Bedienzeit von S_i
- die mittlere Wartezeit pro Besuch ist W_i

$\Rightarrow$ mittlere Verweilzeit an Ressource i pro Anfrage:

$$R'_i = V_i \cdot (S_i + W_i) = D_i + Q_i$$

Konzept der Warte- und Bedienzeiten an CPU und Platte



$\Rightarrow$ in den folgenden Abschnitten:

Berechnung der mittleren Bedienzeiten an unterschiedlichen Komponenten

2.3.2 Bedienzeiten und -anforderungen

Sei i eine Ressource in einem C/S-System,

wenn i von einer Transaktion im Mittel V_i mal besucht wird und jeder Besuch im Mittel S_i Zeiteinheiten dauert, so gilt

$$D_i = V_i \cdot S_i$$

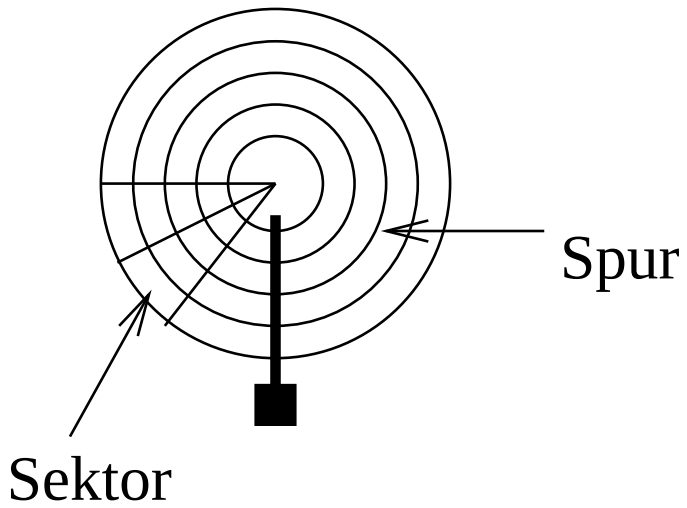
Bestimmung von S_i für verschiedene Ressourcen soll untersucht werden

Plattenlaufwerke

Zugriffszeiten deutlich langsamer als Zugriffszeiten auf Hauptspeicher

deshalb realistische Berechnung der Zugriffszeit wichtig

Struktur einer Festplatte

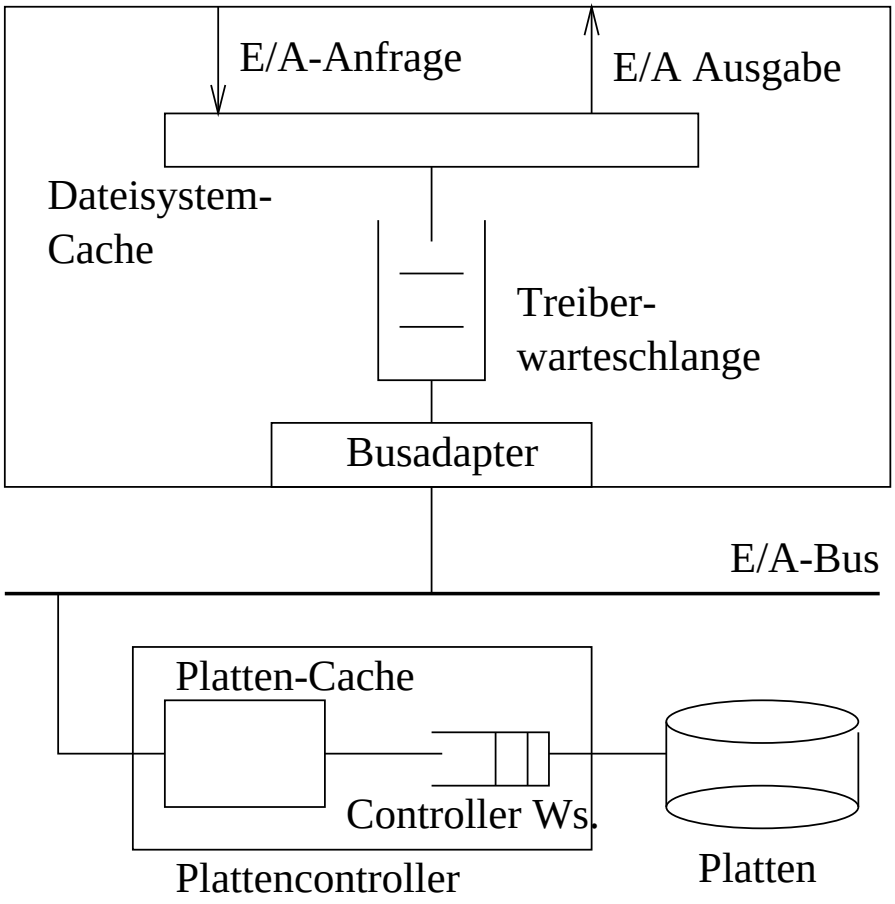


Zugriffszeit unterteilt sich in

- Zeit zur Positionierung des Schreib-/Lesekopfs (*Suchzeit*)
- Zeit bis der Sektor unter dem Lesekopf ist (*Latenzzeit*)
- Zeit zur Datenübertragung (*Transferzeit*)

Heutige Ein-/Ausgabegeräte sind deutlich komplexer, da sie mit Caches auf unterschiedlichen Ebene arbeiten

Struktur eines E/A-Systems



Ablauf einer E/A-Anfrage

- Übertragung der Anfrage an E/A-System
- Cache enthält Menge der Blöcke, auf die vor kurzer Zeit zugegriffen wurde
- falls angeforderter Block im Cache, dann auslesen aus dem Cache, kein Plattenzugriff
- ansonsten wird die Anfrage Plattentreiber geschickt
- dieser schickt die Anfrage zum Plattencontroller
- es wird im Platten Cache nachgeschaut, ob Block vorhanden, ansonsten muss Plattenzugriff erfolgen
- Platten Caches arbeiten mit pre-fetching, d.h. konsekutive Blöcke werden gelesen und im Cache gespeichert
- sowohl Plattencontroller als auch Plattentreiber optimieren die Zugriffe (d.h. ändern u.U. die Reihenfolge von E/A Anfragen)

Zur Bestimmung der Bedienzeit müssen folgende Parameter einbezogen werden

- $\overline{S}_d$ durchschnittliche Zugriffszeit auf Controller und Platte
- $SeekT$ durchschn. Suchzeit (in Sek.)
- $Seek_{rand}$ durchschn. Suchzeit (in Sek.) einer Anfrage, die auf eine beliebige Stelle der Platte zugreift (aus der Plattenbeschreibung entnehmen)
- $DiskSpeed$ Zahl der Umdrehungen pro Minute (aus der Plattenbeschreibung entnehmen)
- $DiskRevolutionT$ Zeit für eine Umdrehung in Sek. ($= 60/DiskSpeed$)
- $RotationalL$ durchschn. Zeit von der Positionierung des Kopfes bis zum Start der Datenübertragung
- $BlockSize$ in Bytes
- $TransferR$ Übertragungsrate in MB/sek
- $TransferT$ Übertragungszeit eines Blockes von der Platte zum Controller
- $ControllerT$ Zeit (in Sek.), die der Controller zur Bearbeitung einer Anfrage benötigt (inkl. Test, ob Block im Cache und evtl. lesen aus Cache)
- P_{miss} W. Block nicht im Cache der Platte

Die durchschnittliche Zugriffszeit lautet

$$\overline{S}_d = ControllerT + P_{miss} \cdot (SeekT + RotationalL + TransferT)$$

Transferzeit berechnet sich als

$$TransferT = \frac{BlockSize}{10^6 \cdot TransferRate}$$

restlichen Größen hängen von der übermittelten Last ab

Last ist definiert als Sequenz von Blöcken

Wir unterscheiden

- zufällige Last (Blöcke in beliebiger Reihenfolge)
- sequentielle Last enthält Sequenzen konsekutiver Blocknummern

Beispiele:

$10, 201, 15, 1023, 45, 39, 782$ ist ein zufällige Last

$4, 350, 351, 352, 353, 80, 104, 105, 106, 107, 108, 243$ ist eine sequentielle Last mit den Sequenzen $(350, 351, 352, 353)$ und $(104, 105, 106, 107, 108)$

Für zufällige Last gilt:

$$P_{miss} = 1$$
$$RunLength = 1$$
$$SeekT = Seek_{rand}$$
$$RotationalL = 0.5 \cdot DiskRevolutionT$$

Für sequentielle Last gilt:

$$P_{miss} = 1/RunLength$$

$$P_{miss} \cdot SeekT = Seek_{rand}/RunLength$$

$$P_{miss} \cdot RotationalL = DiskRevolutionT \cdot \frac{1/2+(RunLength-1)[(1+U_d)/2]}{RunLength}$$

Herleitung:

wenn konsekutive Blöcke in den Cache gelesen werden,

- wird nur der erste Block nicht im Cache gefunden
- muss nur für den ersten Block gesucht werden

Sei $NoReq$ die Anzahl der Blöcke und $NoRuns$ die Anzahl der Sequenzen, dann gilt

$$RunLength = NoReq/NoRuns$$

und damit folgen die ersten beiden Gleichungen

Zur Herleitung der letzten Gleichung

- neue Anforderung kommt an, wenn vorherige abgearbeitet $\Rightarrow$ im Durchschnitt 1/2 Umdrehung
- neue Anforderung kommt an, bevor vorherige abgearbeitet, dann geht 1 Umdrehung verloren

U_d ist die Auslastung der Platte, ankommende Anfrage sieht mit W. U_d den Vorgänger (Annahme!)

Damit lautet die mittlere Zahl von Umdrehungen

$$U_d + 0.5 \cdot (1 - U_d) = (1 + U_d)/2$$

Erster Block einer Sequenz benötigt im Mittel immer
1/2 Umdrehung

womit gilt

$0.5 + (RunLength - 1) \cdot (1 + U_d)/2$ Umdr. pro Seq.

Damit ergeben sich folgende Formeln

Zufällige Last:

$$\overline{S}_d = ControllerT + Seek_{rand} + \frac{DiskRevolutionT}{2} + TransferT$$

Sequentielle Last:

$$\overline{S}_d = ControllerT + \frac{Seek_{rand} + TransferTime}{RunLength} + \frac{[0.5 + (RunLength - 1)(1 + U_d)/2] \cdot DisRevolutionT}{RunLength}$$

Problem $U_d = Ankunftsrate \cdot \overline{S}_d$

wird aber zur Berechnung von $\overline{S}_d$ benötigt

$\Rightarrow$ iterativer Ansatz

U_d vorgeben,

$\overline{S}_d$ berechnen,

daraus neues U_d berechnen,

bis Fixpunkt erreicht

(hier auch möglich: S_d einsetzen und nach S_d auflösen $\Rightarrow$ direkte Berechnung von $\overline{S}_d$)

Beispiel:

Platten eines DB Servers sollen untersucht werden
Folgende Daten sind gegeben

▷ Ankunftsrate 20 Req/sek

▷ 20% zufällig

▷ 80% sequentiell

▷ Blockgröße 2048 Byte

▷ mittlere Sequenzlänge 24

▷ Platten mit 7200 Umd./min

▷ $Seek_{rand}$ 9 msek

▷ Transferrate 20MB/sek

▷ Controller Zeit 0.1 msek

Mittlere Zeit für zufällige Anforderungen:

$$SeekT = 9msek$$

$$RotationalL = 1/2 \cdot 60000/7200 = 4.17msek$$

$$TransferT = 2048/(10^6 \cdot 20) \cdot 1000 = 0.1msek$$

$$\overline{S}_d = 0.1 + 9 + 4.17 + 0.1 = 13.4msek$$

Approximation für Auslastung

$$U_d = 0.02 \cdot (0.1 + 9 + 4.17 + 0.1) = 0.27$$

Mittlere Zeit für sequentieller Anforderungen:

$$P_{miss} = 1/24 = 4.2\%$$

$$SeekT = 9/24 = 0.38msek$$

$$RotationalL = \frac{1/2+23(1+0.27)/2}{24} \cdot \frac{60 \cdot 1000}{7200} = 5.25msek$$

$$\overline{S}_d = 0.1 + 0.38 + 5.25 + 0.042 \cdot 0.1 = 5.73msek$$

Für die mittlere Verweilzeit zufälliger und sequentieller Anforderungen ergibt sich damit

$$\overline{S}_d = 0.2 \cdot 13.4 + 0.8 \cdot 5.73 = 7.26msek$$

mit diesem Wert kann U_d neu berechnet werden und daraus $\overline{S}_d$ für sequentielle Jobs

der Fixpunkt liegt bei $7.02msek$

Plattenarrays

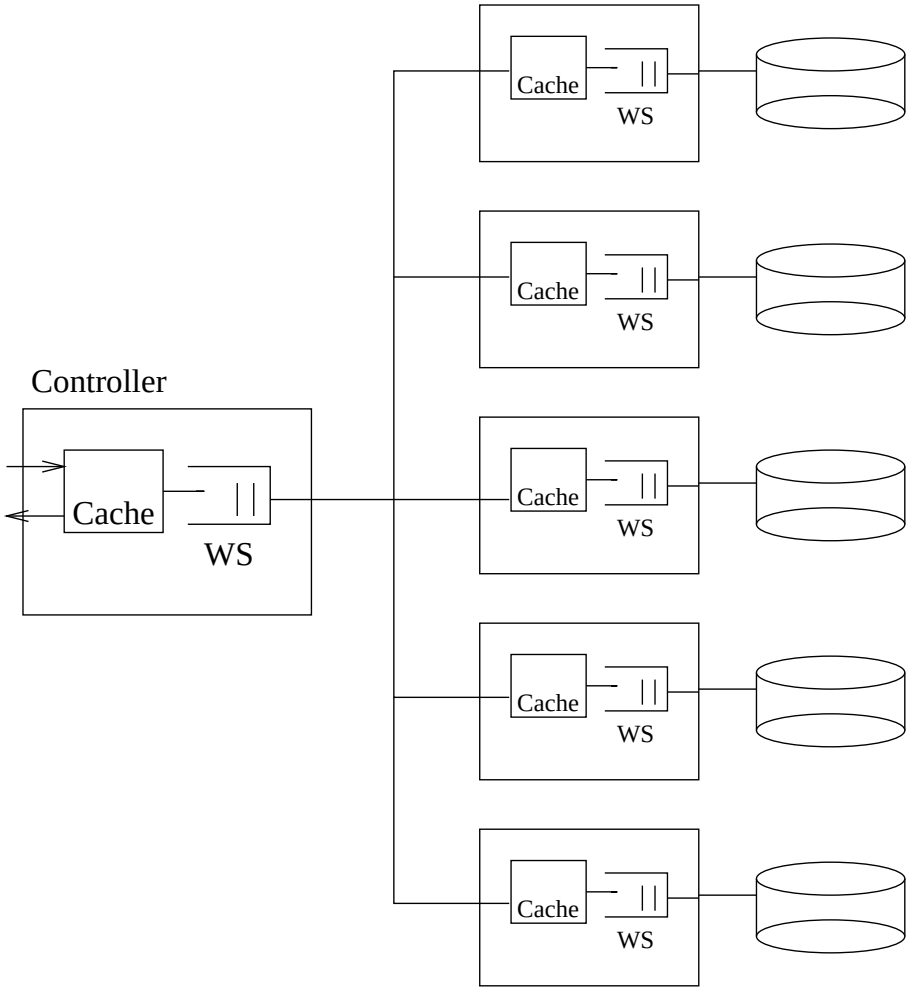
in letzter Zeit immer weiter verbreitet

redundant arrays of independent disks (RAIDs)

Vorteile

- schnellere Zugriffszeit
- Fehlertoleranz

RAID 5 System (hier mit 5 Platten)



Dateien werden über $N-1$ Platten verteilt,
N-te Platte enthält den Paritätsblock $\Rightarrow$ beim Ausfall einer Platte kann jede Datei rekonstruiert werden

Verteilung von Records $A-E$ auf ein RAID5 System mit 5 Platten

Platte 0	Platte 1	Platte 2	Platte 3	Platte 4
PA	A1	A2	A3	A4
B1	PB	B2	B3	B4
C1	C2	PC	C3	C4
D1	D2	D3	PD	D4
E1	E2	E3	E4	PE

jedes Record wird in 4 Streifen (*stripes*) und
einen Paritätsblock unterteilt

Paritätsblöcke werden zyklisch verschoben gespeichert

<u>Verhalten von RAID5 mit 5 Platten</u>

- 20% mehr Speicherplatzbedarf
- Paralleles Lesen großen Anforderungen über 4 Platten
- Paralleles Lesen kleiner Anforderungen über alle Platten
- Paralleles Schreiben großer Anforderungen über alle Platten
- Schreiben kleiner Blöcke erfordert 2 Lese- und 2 Schreiboperationen,
  - ▷ ursprünglichen Streifen und Paritätsblock lesen,
  - ▷ neuen Paritätsblock berechnen,
  - ▷ neuen Paritätsblock und Streifen schreiben

Einige Notationen

- $StripeUnit$ Größe eines Streifens in Byte
- $StripeGroupSize$ Größe der ganzen Gruppe in Byte (Vielfaches von $StripeUnit$)
- n_r Anzahl Streifen, die ein Lesebefehl liest
- n_w Anzahl Streifen, die ein Schreibbefehl modifiziert
- λ_{arr}^r Ankunftsrate von Lesebefehlen am Plattenarray
- λ_{arr}^w Ankunftsrate von Schreibbefehlen am Plattenarray
- λ_{disk}^r Ankunftsrate von Lesebefehlen an einer Platte
- λ_{disk}^w Ankunftsrate von Schreibbefehlen an einer Platte
- N Anzahl Platten im Array
- S_{arr}^r mittlere Bedienzeit eines Lesebefehls
- S_{arr}^w mittlere Bedienzeit eines Schreibbefehls

Lesebefehl generiert Leseanfrage für n_r Platten

Anzahl der Möglichkeiten n_r aus N auszuwählen

$$\left(\begin{array}{c} N \\ n_r \end{array} \right) = \frac{N!}{(N-n_r)!n_r!}$$

Jede Platte gehört zu

$$\left(\begin{array}{c} N-1 \\ n_r-1 \end{array} \right) = \frac{(N-1)!}{(N-n_r)!(n_r-1)!}$$

Gruppen der Größe n_r

bei Gleichverteilung ergibt sich Wahrsch. eines Zugriffs auf eine beliebige Platte zu n_r/N

Zur Paritätsberechnung generieren Schreiboperationen zusätzliche Leseoperationen

Sei $rw(n_w)$ Anzahl Leseoperationen für n_w Schreiboperationen

n_w	1	2	3	4
$rw(n_w)$	2	2	1	0

Rate Leseanforderungen für eine Platte

$$\lambda^r_{disk} = \frac{n_r}{N} \cdot \lambda^r_{arr} + \frac{rw(n_w)}{N} \cdot \lambda^w_{arr}$$

Rate Schreibanforderungen für eine Platte

$$\lambda^w_{disk} = \frac{n_w+1}{N} \cdot \lambda^w_{arr}$$

Bedienzeit einer Leseanforderung für n_r Blöcke

$$S^r_{arr} = \max_{i=1}^{n_r} \{R^r_{disk\,i}\}$$

Bedienzeit einer Schreibanforderung für n_r Blöcke

$$S^w_{arr} = \max_{i=1}^{rw(n_w)} \{R^r_{disk\,i}\} + \max_{i=1}^{n_w+1} \{R^w_{disk\,i}\}$$

Werte für $R^w_{disk\,i}$ und $R^r_{disk\,i}$ aus Analyse der einzelnen Platte

Ein Beispiel: Medizinisches Informationssystem auf RAID5

Dateigröße 80KB pro Röntgenbild

Daten der Platten:

$Seek_{rand} = 9msek$,

7200Umdr./min.,

Transferrate 20Mb/sek,

$StripeUnit 16KB$,

$StripeGroupSize\ 64KB$

$\Rightarrow$pro Anforderung 1.25req

(Anforderungen verhalten sich wie eine zufällige Last)

$S_d = 0.009 + 0.5 \cdot 60/7200 + 16384/20000000 = 0.014sek$

Es gibt nur Leseoperationen:

50% greifen auf 4 Platten zu,

50% greifen auf eine Platte zu

$\lambda_d^r = 0.8 \cdot \lambda_{array}^r/2 + 0.2 \cdot \lambda_{array}^r/2 = \lambda_{array}^r/2$

$U_d = \lambda_d^r \cdot S_d = \lambda_{array}^r \cdot S_d/2 = 0.007 \cdot \lambda_{array}^r$

$\Rightarrow$maximale Zahl an Röntgenbildern wird erreicht,

wenn $U_d = 1$

$\Rightarrow \lambda_{array}^r = 1/0.007 = 142.8$

$\Rightarrow 114.29$ Bilder/sek.,

da ein Bild 1.25 Lesezugriffe erfordert

Netzwerke

Verzögerungen in Netzwerken entstehen durch

- Verarbeitung im Protokoll Stack (TCP/UDP,IP, Ethernet/Token Ring)
- Übertragung durch das Netz
- Weiterleitung in Routern

jede Protokollebene tauscht *Protocol Data Units (PDUs)* aus und fügt den Daten einen Header hinzu

auf Netzwerkebene heißt maximale Größe von Paketen *Maximum Transmission Unit (MTU)*

Paketgrößen einiger Protokolle

Protokoll	PDU-Name	Max. PDU Größe in Bytes	Overhead in Bytes
TCP	segment	65535	20
UDP	datagram	IP datagram	8
IP v4	datagram	65535	20
IP v6	datagram	65535	40
ATM	cell	53	5
Ethernet	frame	1518	18
Token	frame	4472	28
FDDI	frame	4500	28

Overhead addiert sich über die Ebenen

- Router, die Pakete von einem Netztyp zu einem anderen übertragen, müssen Pakete u.U. fragmentieren
- Fragmentierte Pakete werden erst beim Empfänger zusammengesetzt

Empfehlung beim IP Standard:

- erste minimale MTU entlang eines Pfades ermitteln
- dann Datagramm Größe danach wählen

Ausgehend von dieser Empfehlung Ermittlung der Leistungsgrößen

Parameter:

- $MessSize$ Länge der Nachricht
- MTU_n Länge der MTU in Netz n
- $TCPOv$ Overhead durch TCP
- $IPOv$ Overhead durch IP
- $FrameOv_n$ Overhead durch Frames im Netz n
- $Bandw_n$ Bandbreite Netz n
- N Anzahl Netze zwischen Sender und Empfänger

$NDatag$ Anzahl IP Datagramme, die gesendet werden müssen, um Fragmentierung zu vermeiden

$$NDatag = \left\lceil \frac{MessSize+TCPOv}{\min_{n=1}^{N}(MTU_n - IPOv)} \right\rceil$$

Overhead in Netz n:

$$Ov_n = TCPOv + NDatag \cdot (IPOv + FrameOv)$$

Bedienzeit in Netz n:

$$ServT_n = \frac{8 \cdot (MessSize + Ov_n)}{10^6 \cdot Bandw_n}$$

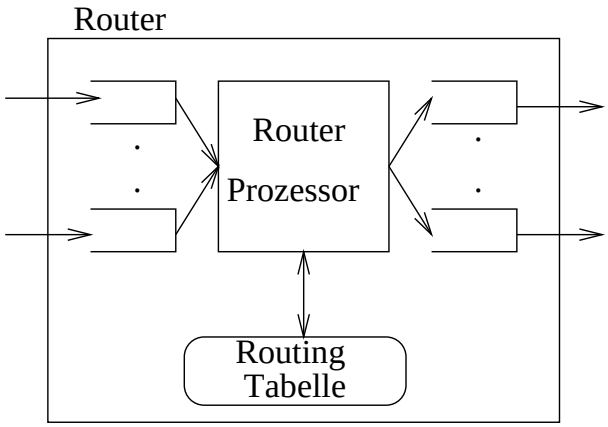
Sei λ_j die Ankunftsrate von Typ j Nachrichten und
$ServT_n^j$ die Bedienzeit von Nachrichten
vom Typ j in Netz n

Auslastung von Netz n:

$$U_n = \sum_{Typ\ j} \lambda_j \cdot ServT_n^j$$

Router

Router bestimmen die Ausgangsleitung auf der
empfangende Pakete weitergeleitet werden



Die Router-Latenzzeit ($RouterLT$) wird pro Paket
in Millisekunden angegeben

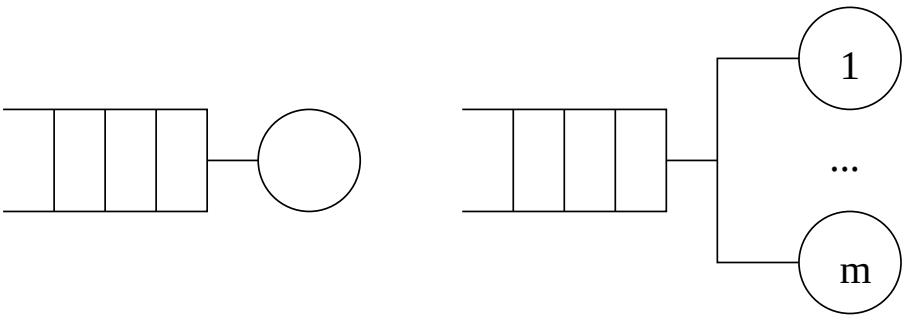
Damit ergibt sich die Bedienzeit

$$RouterServT = NDatag \cdot RouterLT$$

2.3.3 Warteschlangen

Warteschlangen (queues) spalten sich auf

- in den Warteraum (waiting queue) und
- den Bediener (resource/server)



es können 1 bis m (identische) Bediener vorhanden sein

Systeme, bei denen jeder Auftrag immer direkt einen Bediener zugeteilt bekommt bezeichnet man als Verzögerungsstationen (delay resources)

$\Rightarrow$ keine Wartezeiten

Einige Notationen und erste Zusammenhänge

- V_i mittlere Anzahl Besuche Warteschlange i
- S_i mittlere Bedienzeit pro Besuch an i
- W_i mittlere Wartezeit pro Besuch an i
- R_i mittlere Verweilzeit an i
  es gilt $R_i = W_i + S_i$

λ_i Ankunftsrate an i

Annahme für alle Analysen:

Flussgleichgewicht

d.h. System wird lange genug beobachtet, so dass

die Anzahl der Ankünfte
der Anzahl der Abgänge entspricht

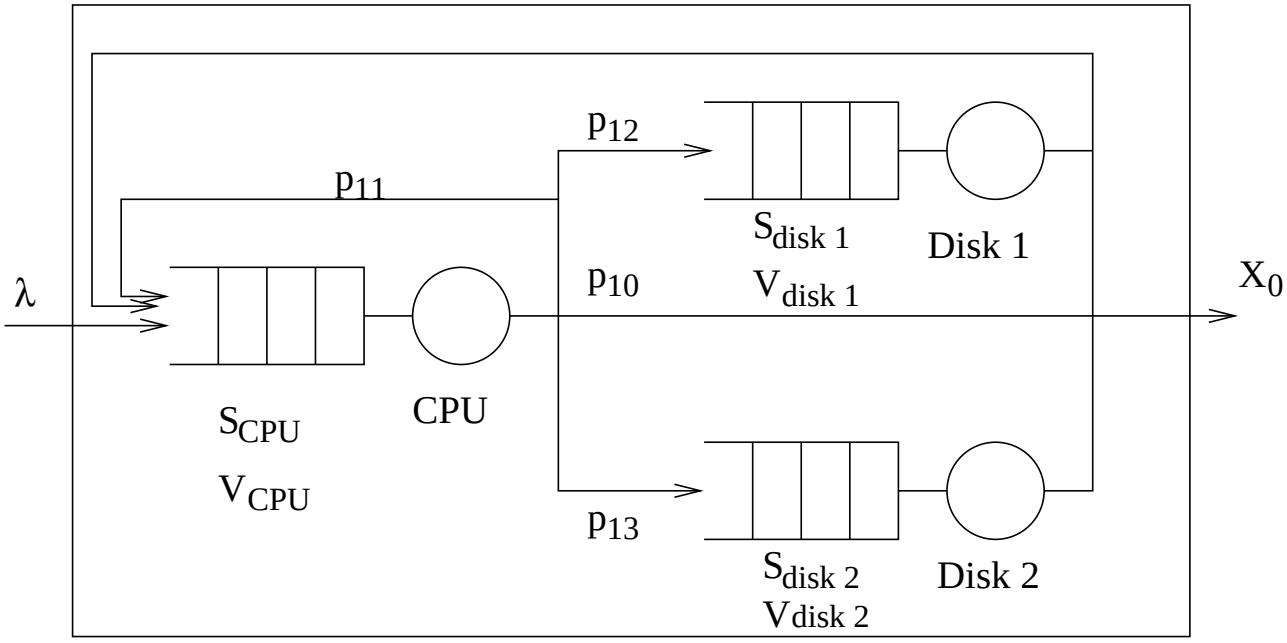
dies setzt voraus: $\lambda_i < 1/S_i$ (bei einem Bediener)

damit gilt für den Durchsatz von i: $X_i = \lambda_i$

Weitere Notationen

- X_0 Systemdurchsatz
- N_i^w mittlere Anzahl wartender Transaktionen an i
- N_i^s mittlere Anzahl Transaktionen in Bedienung an i
  falls i nur eine Ressource beinhaltet, so gilt
  $0 \leq N_i^s \leq 1$ entspricht der mittleren Auslastung
- N_i mittlere Anzahl Transaktionen an i
  es gilt $N_i = N_i^w + N_i^s$
- D_i mittlere Bedienanforderung an i
  (bei allen Besuchen der Transaktion an i)

Beispielmodell



einfaches Modell eines Web Servers

Es gilt:

$p_{10} + p_{11} + p_{12} + p_{13} = 1.00$

Berechnung der V_j aus den p_{ij} durch Lösung eines linearen Gleichungssystems

$$\begin{aligned} V_{CPU} &= p_{11}V_{CPU} + V_{disk\,1} + V_{disk\,2} + 1 \\ V_{disk\,1} &= p_{12}V_{CPU} \\ V_{disk\,2} &= p_{13}V_{CPU} \end{aligned}$$

später mehr zur Berechnung

2.3.4 Regeln und Gesetze zur Leistungsanalyse

Basis der Berechnungen: Operationale Analyse

Auslastungsgesetz

betrachten wir erst einmal den Fall eines Bedieners

U_i Auslastung des Bedieners

(d.h. Anteil der Zeit, in dem der Bediener arbeitet)

wenn wir Bediener T Sekunden beobachten und er war

B_i Sekunden belegt, so gilt $U_i = B_i/T$

andere Beobachtung C_0 Transaktionen werden in

T Sekunden abgearbeitet (d.h. $X_i = C_0/T$), dann gilt

$U_i = B_i/T = (B_i/C_0) \cdot X_i = X_i \cdot S_i$

Im Gleichgewicht gilt $X_i = \lambda_i$ und

$U_i = X_i \cdot S_i = \lambda_i \cdot S_i$

Beispiel:

Netz mit überträgt 1000 pak/sek

Paketlänge 0.15 msek

Auslastung: $1000 \cdot 0.00015 = 0.15 = 15\%$

Auslastung entspricht mittlerer Population des Bedieners

also $U_i = N_i^s$

Auslastung bei m Bedienern

$$U_i = X_i \cdot S_i/m$$

Flussrelation

Transaktion besucht Warteschlange i
durchschnittlich V_i mal

wenn X_0 Transaktionen pro Zeiteinheit das System verlassen, so besuchen

$$V_i \cdot X_0$$

Transaktionen i pro Zeiteinheit

Damit gilt

$$X_i = V_i \cdot X_0$$

Beispiel:

DB Transaktionen führen 4.5 E/A-Operationen auf DB-Server durch

Bei Beobachtung über eine Stunde wurden 7200 Transaktionen beobachtet

- Wie hoch ist der Durchsatz der Platte?
- Falls eine E/A-Operation durchschnittlich 20 msek benötigt, wie hoch ist die Auslastung der Platte?

$X_0 = 7200/3600 = 2$ tr/sek
$X_d = 4.5 \cdot 2 = 9$ tr/sek
$U_d = X_d \cdot S_d = 9 \cdot 0.02 = 0.18 = 18\%$

Bedienanforderungsgesetz

Bisher definiert $D_i = V_i \cdot S_i$

weitere Möglichkeit

$$D_i = V_i \cdot S_i = (X_i/X_0) \cdot (U_i/X_i) = U_i/X_0$$

Beispiel:

- Web Server wird 10 Minuten lang beobachtet
- CPU war zu 90% ausgelastet während dieser Zeit
- 30000 HTTP Anfragen wurden abgearbeitet

Wie hoch ist der durchschnittliche CPU Zeitbedarf einer HTTP Anfrage?

$$U_{CPU} = 90\%$$

$$X_{Server} = 30000/(10 \cdot 60) = 50 \text{ req/sek}$$

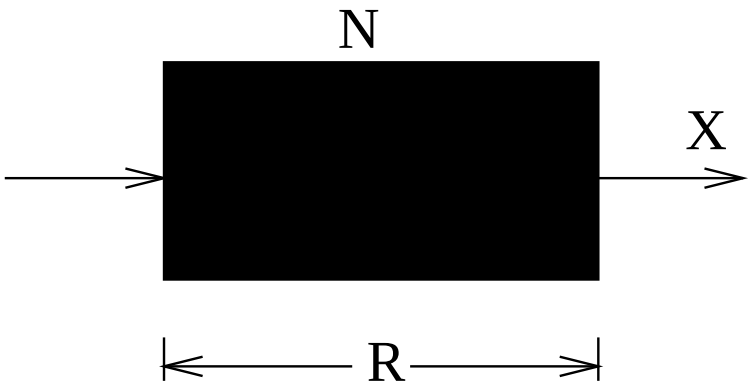
$$D_{cpu} = V_{cpu} \cdot S_{cpu} = U_{cpu}/X_{Server} = 0.90/50 = 0.018 \text{ sek}$$

Gesetz von Little

Einfaches und fundamentales Gesetz der Leistungsanalyse

Im sehr weitem Kontext anwendbar

Basis für viele Analysealgorithmen



Inhalt des schwarzen Kastens (black box) ist unbekannt bzw. irrelevant

Beobachtete Größen

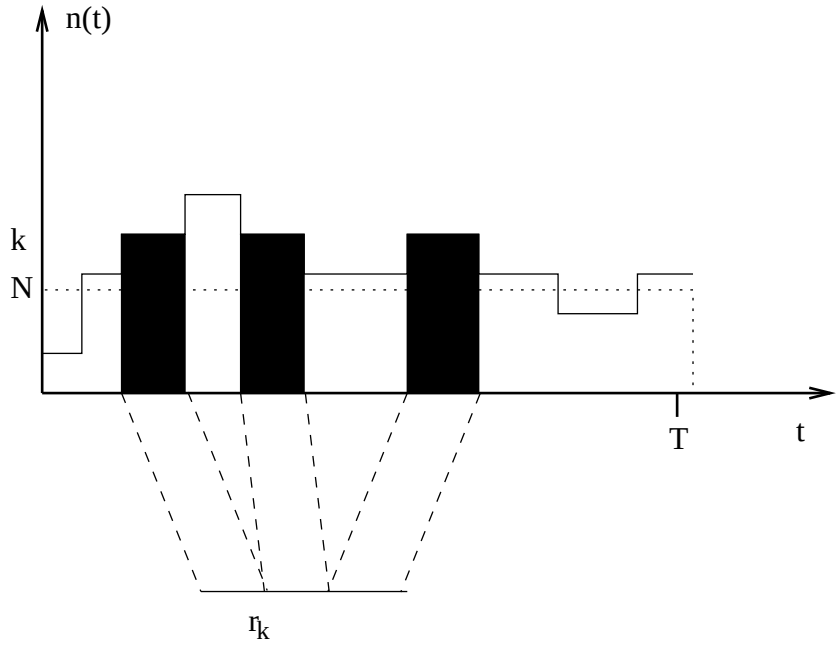
- Kunden bleiben durchschnittlich R sek im Kasten
- N Kunden sind durchschnittlich im Kasten
- X Kunden verlassen durchschnittlich pro Sekunde den Kasten

Folgender Zusammenhang gilt

$$\boxed{N = X \cdot R}$$

Kurze Herleitung

Systemzustand im Beobachtungsintervall $[0, T]$



- $n(t)$ Kundenzahl zum Zeitpunkt t
- f_k prozentualer Anteil am Intervall $[0, T]$ zu dem sich k Kunden im Kasten befinden
- r_k Zeitdauer, zu der sich k Kunden im Kasten befinden

damit gilt $f_k = r_k/T$

$$N = \sum_k k \cdot f_k = \sum_k k \cdot \frac{r_k}{T}$$

- C_0 Gesamtzahl Kunden, die den Kasten im Beobachtungsintervall verlassen haben

$$N = \frac{C_0}{T} \cdot \frac{\sum_k k \cdot r_k}{C_0} = X \cdot R$$

Beispiel 1:

- Beobachtung eines NFS Servers über Zeitraum von 30 Minuten
- 10800 E/A-Operationen während dieser Zeit
- im Durchschnitt 3 E/A-Anforderungen am Server

Wie groß ist die mittlere Verweilzeit einer Anfrage?

$$X = 10800/1800 = 6 \text{ req/sek}$$

$$R = 3/6 = 0.5 \text{ sek}$$

Bisher nicht erläutert, was im schwarzen Kasten ist

und dies ist auch beliebig, womit gilt

$$N_i^w = X_i \cdot W_i$$

$$N_i^s = X_i \cdot S_i$$

$$N_i = X_i \cdot R_i$$

Beispiel 2:

- Router mit Latenzzeit $200\mu sek/pak$
- Übertragungsrate 30000 pak/sek

Wie groß ist die mittlere Paketzahl im Router?

$30000 \cdot 0.0002 = 6$ Pakete

Übersicht über die bisherigen Formeln der Operationalen Analyse

- Auslastung einer Ressource: $U_i = X_i \cdot S_i$
- Flussrelation: $X_i = V_i \cdot X_0$
- Bedienanforderungsgesetz: $D_i = V_i \cdot S_i = U_i / X_0$
- Littles Gesetz $N = X \cdot R$
- Littles Gesetz für Ressource i: $N_i = X_i \cdot R_i$

Kapazitätsplanung mittels Operationaler Analyse?

- nur für bestimmte Fälle
- sehr einfache Modellierung des Bedienanforderungsgesetztes für den Flaschenhals (Bottleneck)

Grundidee des Vorgehens:

- Durchsatz am Flaschenhals bestimmt den Durchsatz des Gesamtsystems
  ($\Rightarrow$ solange Station i Flaschenhals bleibt, kann sich die Betrachtung auf i fokussieren)
- Durchsatz und Auslastung in der IST-Situation messen

Annahmen zur Analyse:

- Flussgleichgewicht: Last = Durchsatz
- aus der Auslastung des Flaschenhalses lässt sich Einhaltung von SLAs ableiten
  (Auslastung des Falschenhalses ist Key Performance Indicator (KPI))

Kapazitätsplanung für zukünftige Entwicklung:

- Durchsatz (= Last) wird mit Wachstumsfaktor multipliziert
- Bedienanforderung wird durch Verbesserungsfaktor dividiert
- Faktoren sind zu prognostizieren
- Einhaltung von SLAs aus den zukünftigen Auslastungen ableiten

Voraussetzungen der Vorgehensweise:

- untersuchte Ressource bleibt Flaschenhals
- Lastprofil bleibt im wesentlichen gleich
- Auslastung bzgl. SLA interpretierbar

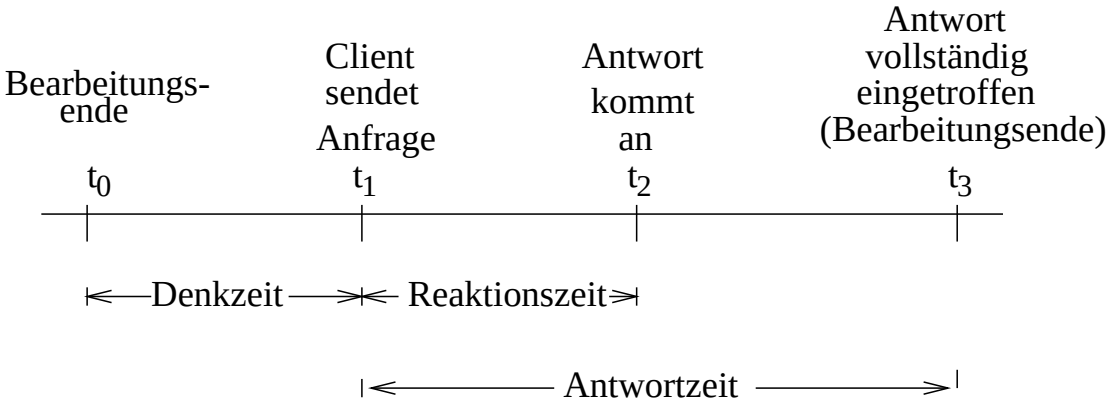
2.3.5 Leistungsmaße in Client-Server Systemen

Wichtigste Metriken zur Bewertung von C/S-Systemen:

- Antwortzeit
- Durchsatz
- Kosten

Ersten beiden wurde intuitiv eingeführt und sollen nun im Kontext interpretiert werden

Ablauf einer Transaktion



Zum Zeitpunkt t_0 wurde eine Anfrage vollständig beantwortet

- Intervall (t_0, t_1) beschreibt Zeit, die der Client benötigt um neue Anfrage zu formulieren (*Denkzeit*)

- Antwort des Servers beginnt zum Zeitpunkt t_2 einzutreffen ($t_2 - t_1$ ist die *Reaktionszeit*)

- Antwort ist zum Zeitpunkt t_3 vollständig eingetroffen ($t_3 - t_1$ ist die *Antwortzeit*)

am Beispiel Web :

- Denkzeit := Zeit zwischen zwei Seitenanforderungen
- Reaktionszeit := Zeit von Anforderung einer Seite bis zum Start des Seitenaufbaus
- Antwortzeit := Zeit von Anforderung der Seite bis zum vollständigen Aufbau

Antwortzeit ist das Maß, welches
für den Benutzer von Interesse ist

Durchsatz ist mehr für Betreiber von Interesse

Einheit des Durchsatzes hängt vom Transaktionstyp ab

- NFS Server E/A-Operationen pro Sekunde
- HTTP Server HTTP Operationen pro Sekunde
- DB Server Transaktionen pro Sekunde

Kosten

Kosten hängen mit Leistung zusammen

also Kosten pro "Leistungseinheit"

TPC (Transaction Processing Council) Benchmark für online Transaktionssysteme misst
$ per Transaktion pro Sekunde

Ergebnis: Preis pro Durchsatzeinheit

Kosten umfassen Hardware und Software

Heutige Client-Server Anwendungen laufen oft als Web-Anwendungen

Elemente einer solchen Umgebung

- Server (Web-sites, proxy, cache, mirror)
  Vielzahl von potenziellen Clients
- Clients (oft Browser, email, ftp)
  Browser kann mit jedem Server mit HTTP Adresse kommunizieren
- Netze (LAN und WAN)
  Umgebung oftmal nicht transparent

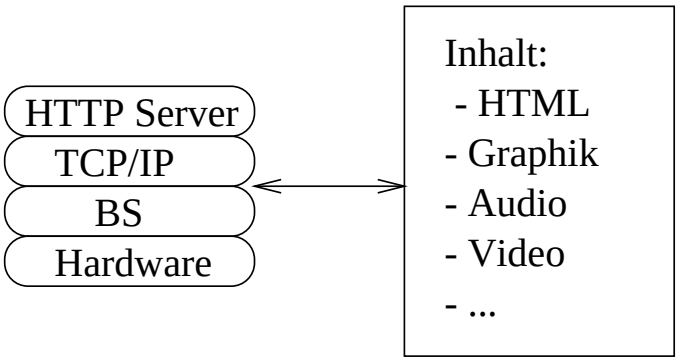
Antwortzeit und Verfügbarkeit sind die kritischen Größe im Web (Benutzer warten nicht lange! Nicht zugreifbare Seiten können nicht besucht werden)

Zusätzlich Probleme:

- hohe Varianz in der Anforderungsströme
- große Unterschiede in Objektgrößen ($10^3 - 10^7$ Bytes)

+ viele Anwendungen, die auf Web-Servern laufen (müssen)

Infrastruktur



Dokumente (Inhalt) beschrieben in HTML oder XML

Darstellung von Text

+ Zeiger auf andere Dokumente

+ Integration von Bildern, Audio, Video ..

Ein Benutzeraufruf kann zu Vielzahl von Aufrufen des Clients beim Server führen !

Nachladen von Bildern, Auflösen von Verweisen etc.

Leistungssteigerung durch Caching, paralleles Laden

Insbesondere Implementierung TCP/IP im BS
beeinflusst HTTP Leistungsfähigkeit

HTTP Protokoll auf der Anwendungsschicht,
welches TCP benutzt

HTTP definiert Anfrage-Antwort Transaktionen
(request-reply, Web-Transaktionen)

Schritte einer HTTP Transaktion:

- Abbildung des Server Namens auf die IP Adresse
- Einrichten der TCP/IP Verbindung
- Übertragen der Anfrage
  (URL + Methode + zus. Informationen)
- Empfangen der Antwort
  (HTML Text oder Bild oder andere Information)
- Schließen der TCP Verbindung

HTTP ist zustandsloses Protokoll
(d.h. jeder Transfer ist isoliert von vorherigen oder
nachfolgenden Anfragen)

Vorteil: Server braucht sich nicht Zustände oder
Identitäten einzelner Clients zu merken

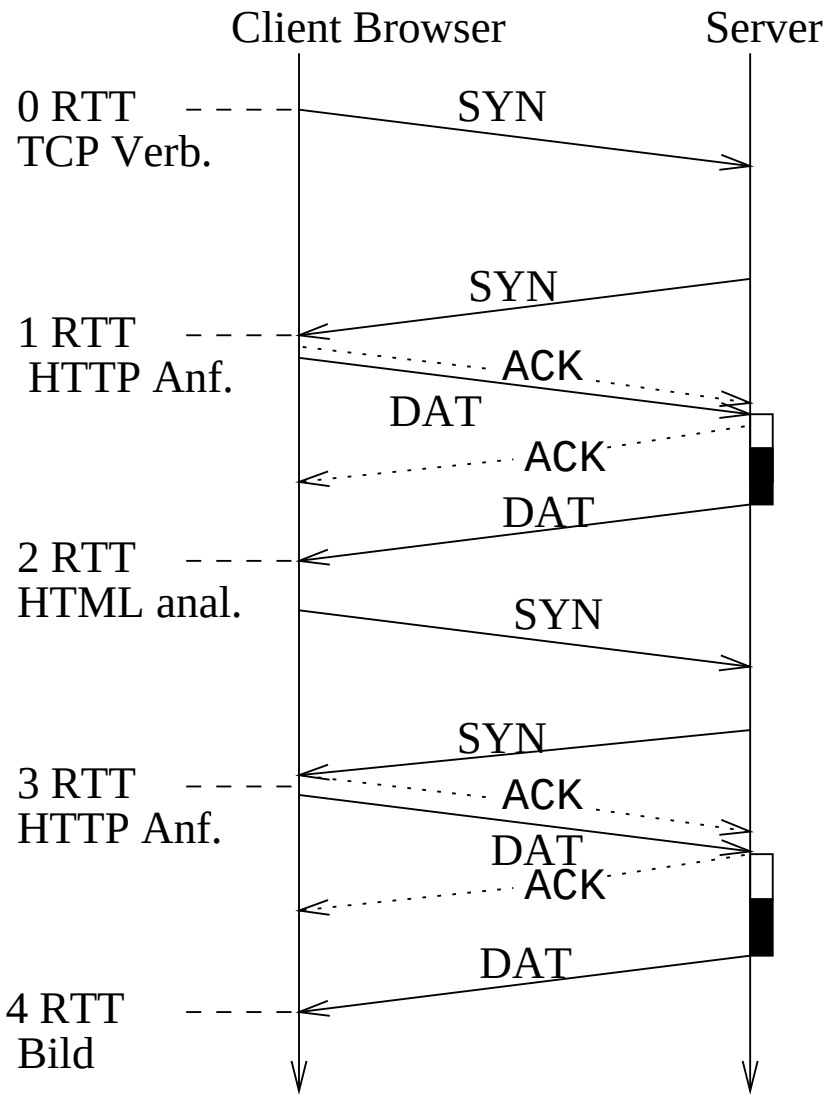
Nachteil: Leistungseinbußen insbesondere in
Originalversion HTTP 1.0
(z.B. Etablierung einer neuen Verbindung
pro Anforderung)

Zeitverzögerungen ergeben sich aus

- Verzögerungen zur Etablierung von Verbindungen
- Verzögerungen zur Übertragung der Daten

Ablauf einer Interaktion

Anforderung HTML Seite + enthaltenem Bild in HTTP 1.0



Notwendige Interaktionen:

- Client öffnet TCP Verbindung
- Client sendet HTTP Anforderung zum Server
- Server
  - führt Aktion aus
  - sendet Daten
  - schließt Verbindung
- Client untersucht HTML Dokument und findet enthaltenes Bild
- Client öffnet TCP Verbindung
- Prozess setzt sich wie beschrieben fort

Konsequenz:

- mind. $4 \cdot RTT$ ist notwendig für HTML Seite mit einem Bild
- jedes weitere Bild erfordert $2 \cdot RTT$ zusätzlich

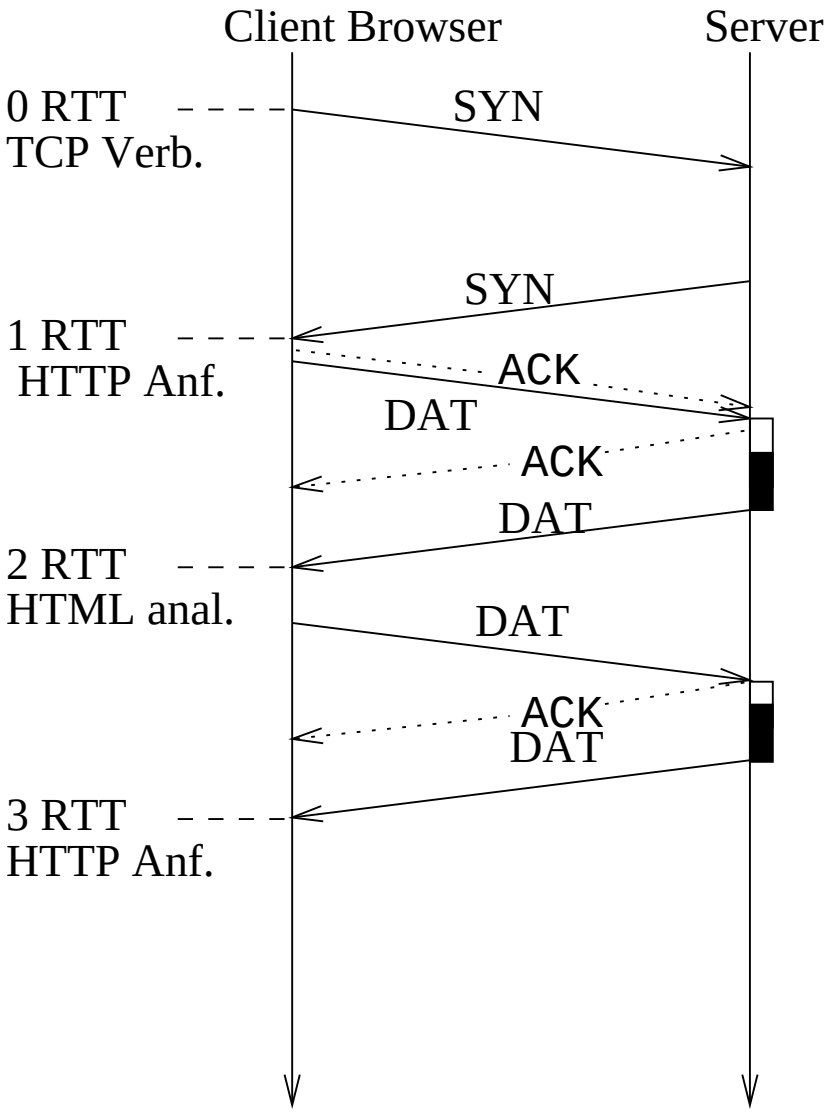
Weiteres Problem: *slow start* von TCP

- geringer Durchsatz am Anfang einer Übertragung
- die meisten HTML Dokumente sind so klein, dass keine Vergrößerung der Fenstergröße erfolgt

$\Rightarrow$ lange Antwortzeiten

Neue Version HTTP 1.1

- mit persistenter Verbindung
- Pipeline von Anforderungen



$\Rightarrow$ Übertragungszeit $3 \cdot RTT$ statt $4 \cdot RTT$

Ziel: Identifikation von detaillierten Verzögerungszeiten einer Web-Transaktion

- Browser
  - Anklicken eines Hyperlinks
  - Browser sucht Dokument im lokalen Cache
  - Falls Dokument nicht im Cache
    * IP Adresse vom DNS holen
    * TCP Verbindung etablieren
    * HTTP Anforderung senden
  - bei Empfang des Dokuments, dieses formatieren und anzeigen
- Netz
  - Übertragung der Daten vom Client zum Server und zurück
- Server
  - Anforderung kommt an
  - Interpretieren der Anforderung
  - Ausführung der angeforderten Methode
  - Falls nötig: Lesen der angeforderten Datei
  - Senden der Datei

Zeiten der einzelnen Schritte

R_r Antwortzeit (zu ermitteln)

R_{Cache} Zeit, um Daten aus dem Cache zu lesen

Falls Daten im Cache gilt $R_r' = R_{Cache}$

Falls Daten nicht im Cache gilt

$$R_r = R_{Browser} + R_{Net} + R_{Server}$$

wobei einzelne Zeiten jeweils Gesamtzeiten
an den Komponenten sind

z.B. $R_{Net} = R_{N1} + R_{N2}$

mit R_{N1} Übertragungszeit vom Client zum Server
R_{N2} Übertragungszeit vom Server zum Client
+ evtl. Übertragungszeiten zum/vom DNS

Allgemein gilt $R_{Cache} \ll R_{Net} + R_{Server}$

Falls N_C von N_T Seiten im lokalen Cache
gefunden werden, so gilt

$$R_r = p_C \cdot R_{Cache} + (1 - p_C) \cdot R_r$$

mit $p_C = N_C / N_T$

Berechnung der einzelnen Werte für $R_.$ wie beschrieben

Flaschenhälse

- Falls die Anzahl der Clients und Server steigt, wird die Leistung für den Benutzer durch die Leistungsfähigkeit einiger Komponenten beschränkt
  $\Rightarrow$ Flaschenhälse (Bottlenecks)
- Leistungsanalyse ist oft Bestimmung des Flaschenhalses
- Falschenhals ist die Komponente deren Leistungsfähigkeit als erstes ausgeschöpft ist
- Leistungssteigerung beim Flaschenhals bringt größten Effekt für Gesamtleistung

Leistungsaspekte

Leistung ähnlich zu bekannten Ansätzen für
Rechensysteme oder Kommunikationssysteme
Aber Web Umgebungen

- sind größer (u.U. mehrere Millionen Clients)
- heterogener in Software und Hardware
- von dynamischer Struktur
- haben größere Variabilität im Verhalten
- erlauben kaum Messungen

Quality of Service

Anforderungen, die Benutzer an einen Dienst haben

Änderungen der Dienstgüte werden sofort bemerkt
(in Realzeit) und führen oft zum Verbindugsabbruch

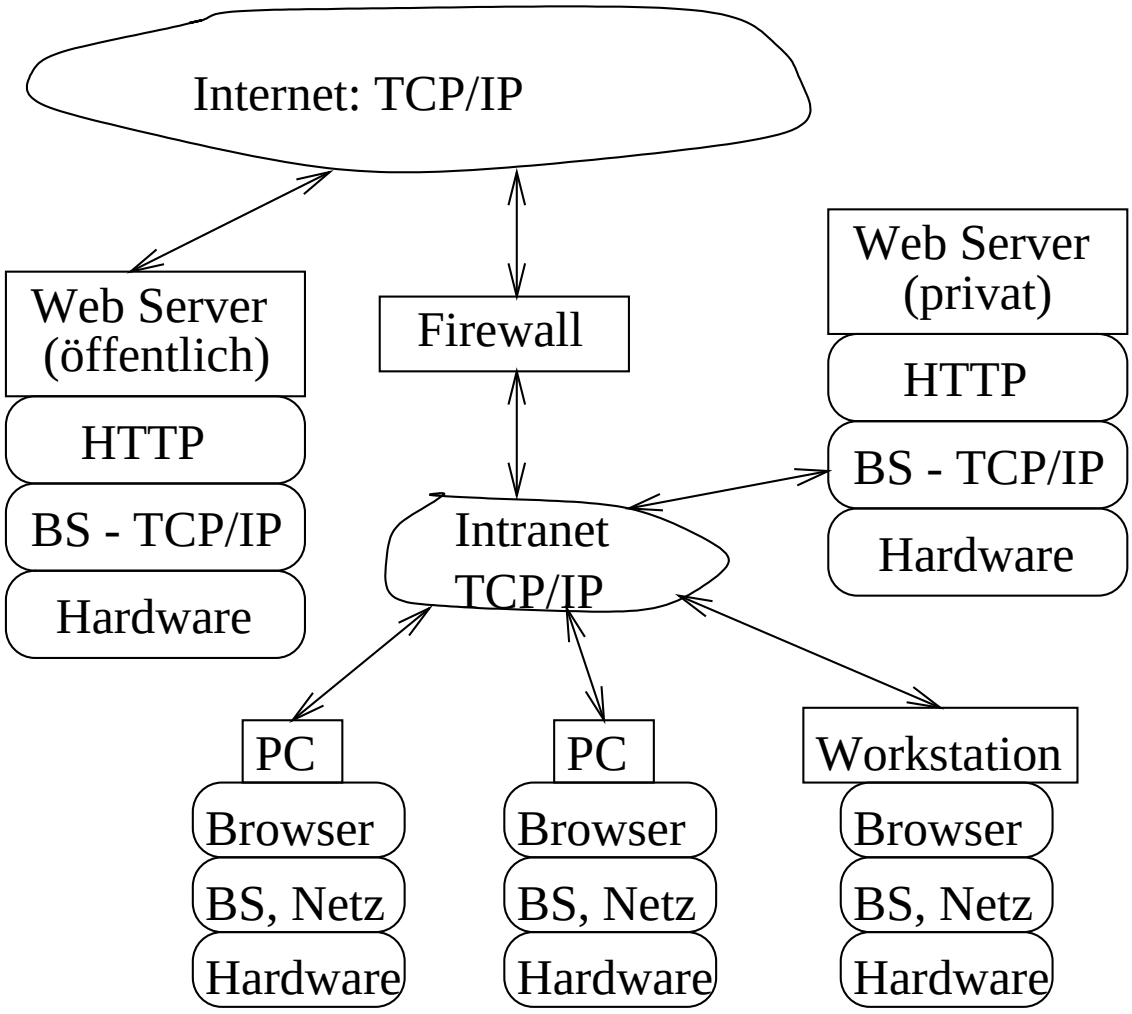
Typische Sichtweisen des Kunden

- Antwortzeit
- Verfügbarkeit
- Zuverlässigkeit
- Vorhersagbarkeit
- Kosten

und nicht Interna der Verbindung oder des Servers

QoS ist auch zu Stoßzeiten sicherzustellen
d.h. Kapazitätsplanung für Hochlastphasen

Infrastruktur des WWWs



Basiskomponenten

- Server
- Browser (auf dem Client)
- Firewalls
- Netze

Firewalls sind

Sicherheitsmechanismen zum Schutz von

- Daten
- Programmen
- Computern

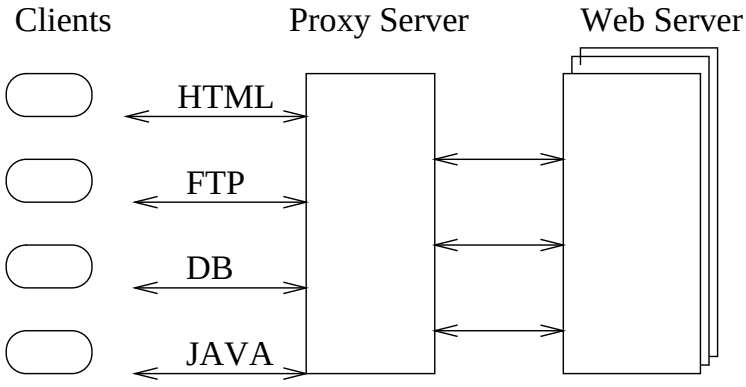
in einem privaten Netz

Firewalls verhindern den Zugriff nicht autorisierter Benutzer von außen auf Netzinterna

Sämtlicher Verkehr zwischen Internet und Intranet läuft über Firewalls

$\Rightarrow$ Firewalls sind potentielle Flaschenhälse

Oft agieren Proxy Server als Firewalls für Web Seiten



Proxy Server ist spezieller Web Server, nämlich gleichzeitig

- Server für die Clients und
- Client für andere Server

Proxy Server empfängt Anforderungen der Clients

- leitet diese weiter an den Server
- nach Empfang der Antwort wird
  diese an den Client weitergeleitet

Oft beinhalten Proxy Server Caches

damit können oft zugegriffene Seiten
vom Proxy Server bedient werden

Besonders effektiv, wenn viele Benutzer mit
ähnlichem Verhalten über einen Proxy Server zugreifen

Effektivität des Caching kann gemessen werden in

- Trefferwahrscheinlichkeit (Hit Ratio)
- gewichtete Trefferwahrscheinlichkeit
  (Gewicht durch Dokumentgröße)
- Volumen aus dem Cache transferierter Daten

Alternative zum Caching auf Client/Proxy Server Seite
ist Spiegelung auf Server Seite

Bewertung auf Basis der Kosten-Nutzen Analyse

- zusätzliche Kosten durch Caches, Spiegelserver
- zusätzlicher Nutzen durch diese Maßnahmen

Neben Auswahl der Hard-/Software ist
Lage der Server entscheident

Web Server Software (HTTP server/daemon) ist ein Programm, welches Datenfluss von/zum Internet/Intranet steuert

Ablauf einer Transaktion

- Etablierung der Verbindung
- Senden der Daten
- Abbau der Verbindung
- Wartezustand für neue Anforderung

Web Server können mehr als eine Anforderung gleichzeitig behandeln durch

- *forking* des HTTP Prozesses (Unix Fork)
- HTTP Programm aus mehreren Threads
- Verteilung der Anfragen unter einen Pool laufender Prozesse

Für Web Server, die mehrere Millionen Anfragen bearbeiten müssen, reicht dies nicht aus

$\Rightarrow$ Verteilung des Servers über mehrere Maschinen

- dynamische HTTP Umleitung
  - Web Server sendet neue URL mit Spiegelserver zum Client
  - Client schickt Anforderung an neuen Server
  - Vorgehen für Benutzer nicht sichtbar
  - führt zu zusätzlicher Last

- *round-robin Domain Name Server* (DNS)
  - erlaubt Server clustering
  - Server Name gehört zu mehreren Servern mit eigenen IP Adressen
  - DNS verschickt zugehörige IP Adressen im round robin Prinzip
  - damit bzgl. Zugriffszahl relativ gleich verteilte Last

- *single IP image*
  - IP Adresse des Servers gehört zu einem TCP Router
  - Router verteilt Anforderungen unter angeschlossenen Servern
  - Server schickt Antwort mit Router Adresse

Lastbeschreibung für Web Server

Bisher wenige Arbeiten über
Lastbeschreibung und -charakterisierung für Web Server

Hier einige erste Charakteristika und Invarianten

Viele Web-Seiten haben dynamische Inhalte
(z.B. interaktive Graphiken)

Programme laufen entweder

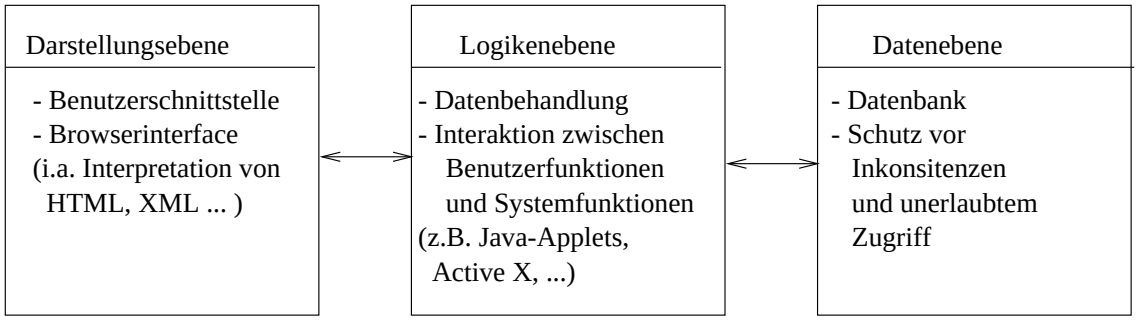
- auf Client Seite
  - Java
  - ActiveX
  - Dynamic HTML
- oder auf Server Seite
  - Common Gateway Interface (CGI)
  - FastCGI Skripte
  - Internet Server API (ISAPI)

CGI Skripten separate Programme außerhalb des Servers
$\Rightarrow$ zus. Last kann Server stark verlangsamen!
(Last wird vom Client erzeugt!)

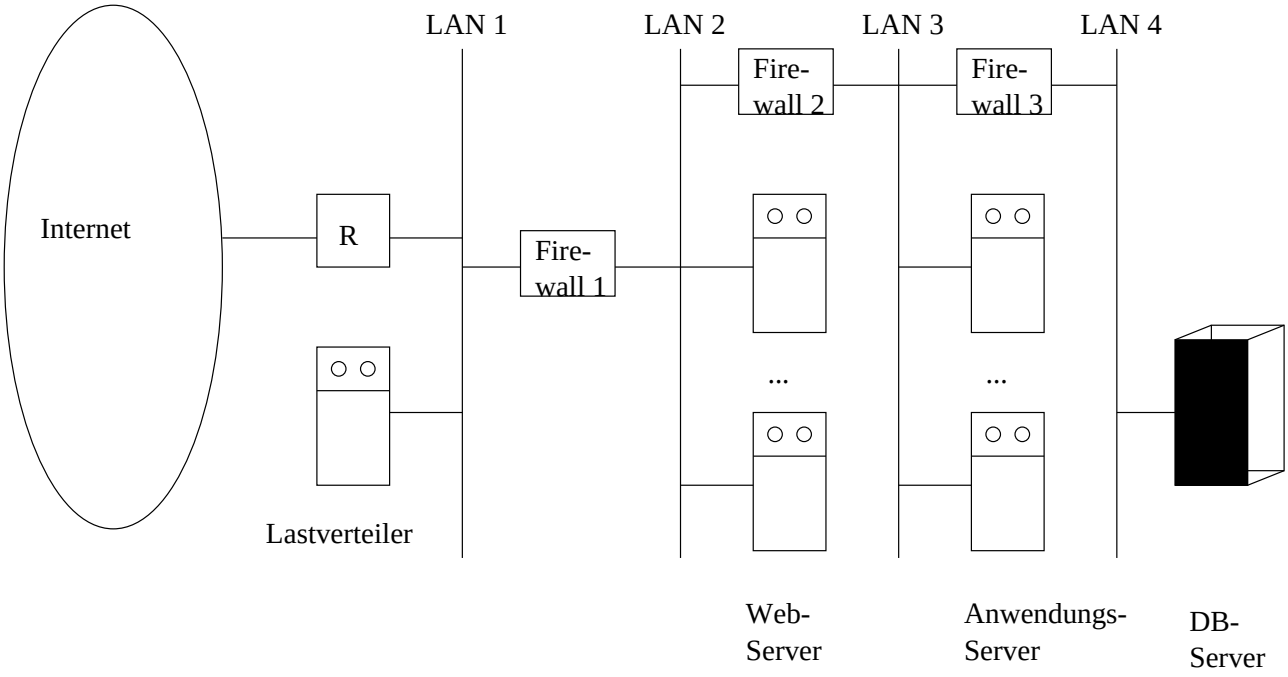
Bei vielen kommerziellen Servern verbrauchen CGI Skripte zur Informationssuche den größten Teil der CPU Zeit

Typischer Aufbau großer Server (z.B. für E-Commerce Anwendungen)

Mehrstufige Architekturen (multi-tier architectur)



Beispielaufbau eines E-Commerce Servers



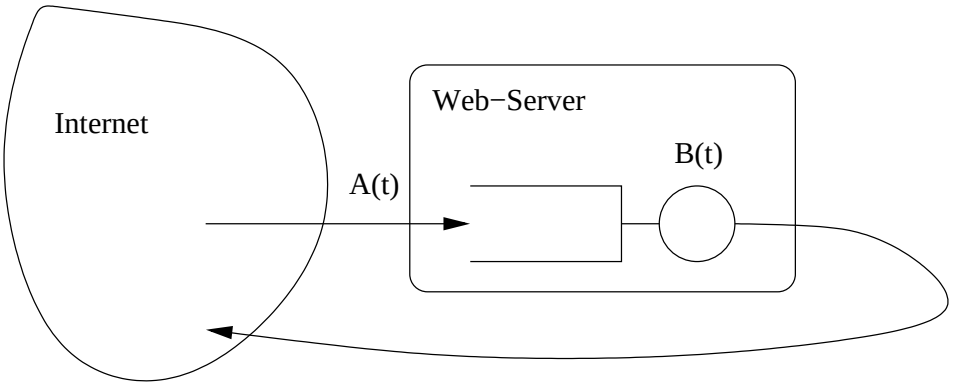
Lastmessung und -modellierung bei Web-Servern

Ein Web-Server speichert eine Menge von Dateien,
auf die Benutzer zugreifen

Zugriffe werden charakterisiert durch

1. Zeit zwischen aufeinander folgenden Zugriffen
2. Dauer eines Zugriffs (d.h. Verteilung der Dateigrößen)
3. Wahrscheinlichkeit auf eine bestimmte Datei zuzugreifen

Einfaches Modell:



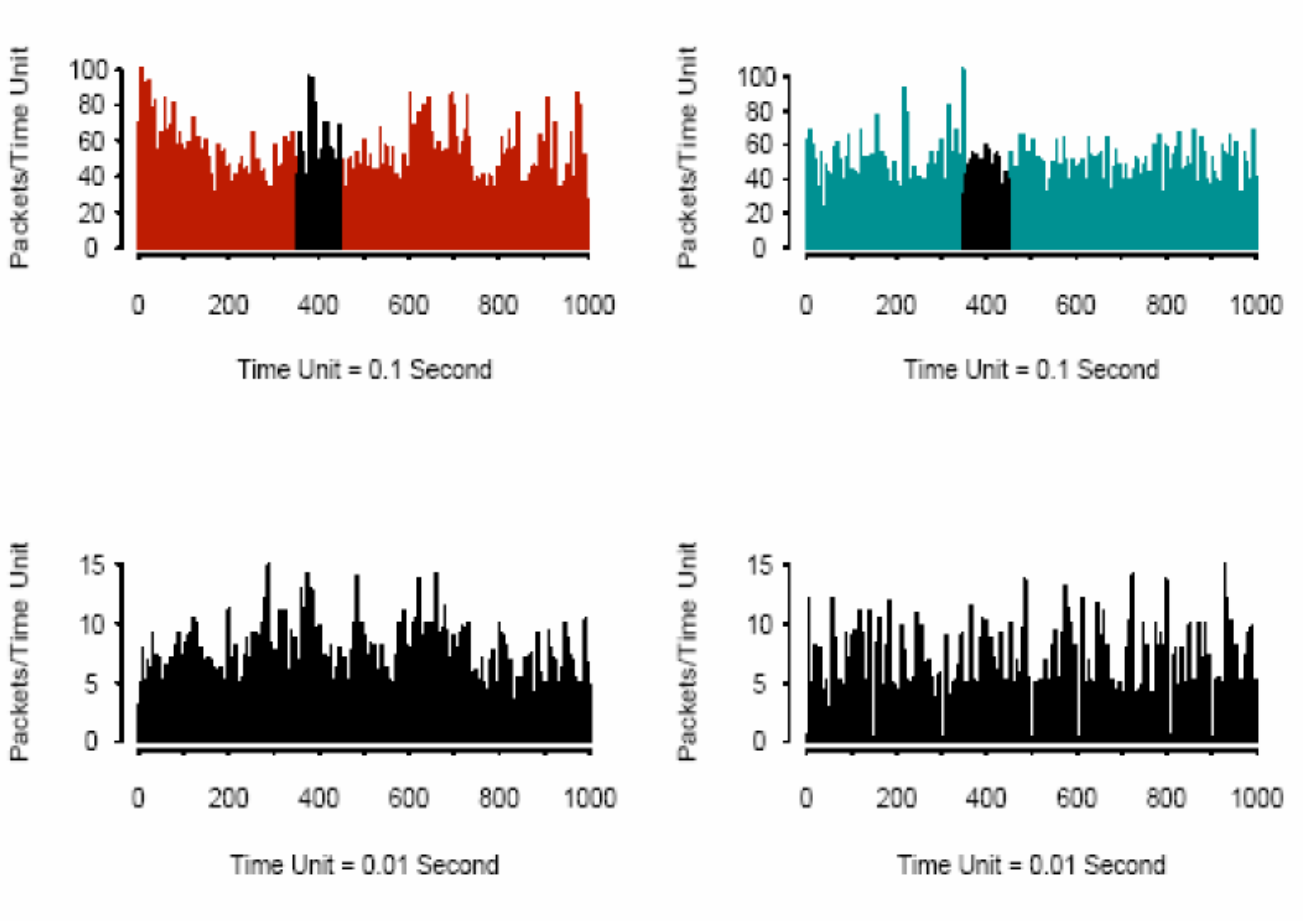
1. bestimmt den Ankunftsstrom A(t),
2. bestimmt die Bedienzeit B(t)
3. wird beim einfachen Modell nicht berücksichtigt,
ist aber bedeutsam für das Caching von Dokumenten

- Also Bestimmung der notwendigen Daten durch Messung (vorhandene Traces, Monitoring)
- Darstellung des beobachteten Verhaltens durch mathematisches Modell (Verteilung, stochastischer Prozess)

Messung von Zwischenankunftszeiten

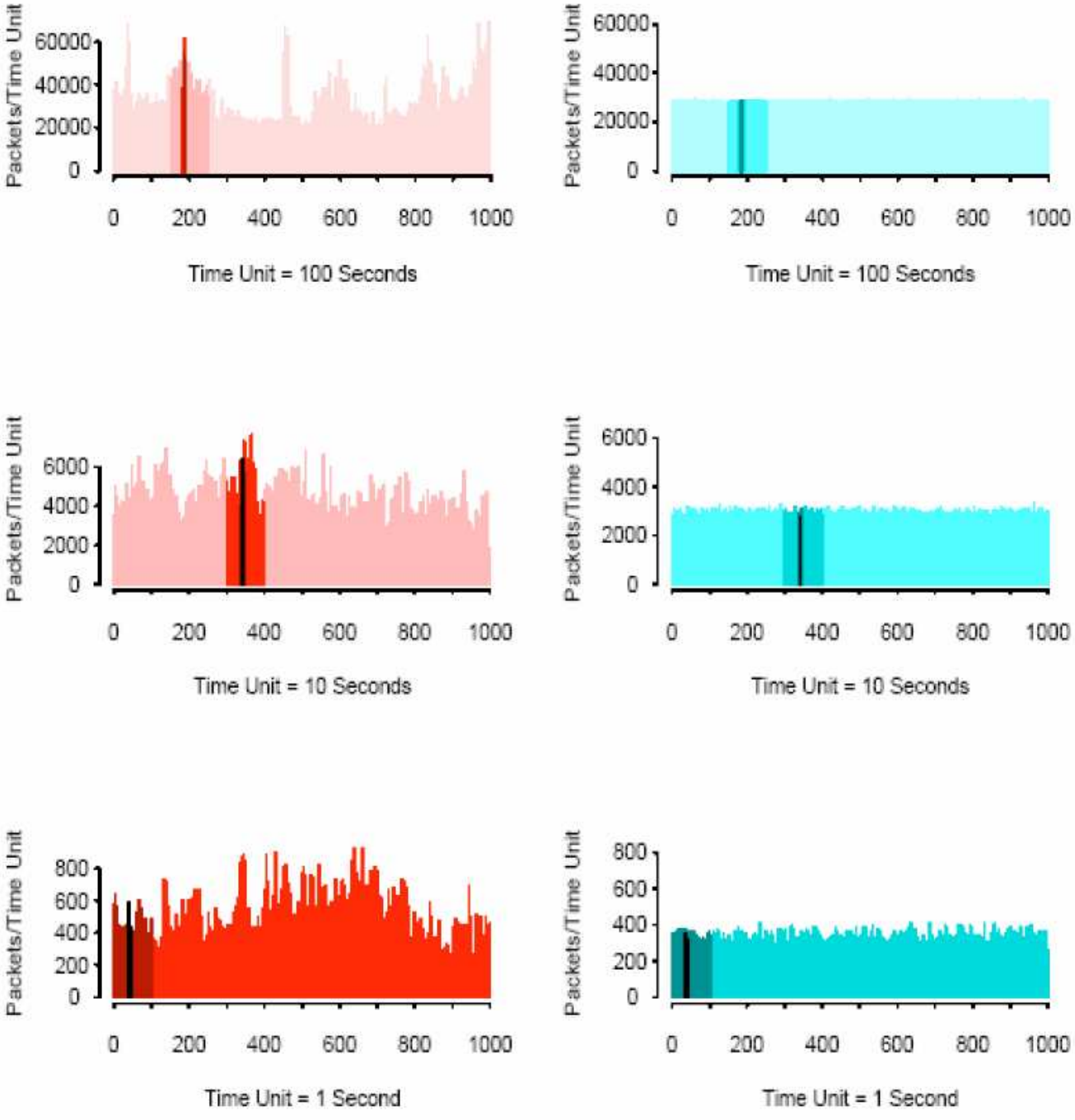
Vergleich von gemessenen Ankunftsraten und modellierten Ankunftsraten (negativ exponentielle Zwiwchenankunftszeiten)

Darstellung auf zwei Zeitskalen
(Ankünfte pro 0.1 und pro 0.01 Sek.)



Keine strukturellen Unterschiede feststellbar

Weitere Aggregierung der Zeitskala



Deutliche Unterschiede sichtbar!

Zeit zwischen Zugriffen hat hohe Varianz

- auf unterschiedlichen Zeitskalen hohe Varianz mit Bildung von sogenannten *bursts*
  - Selbstähnlichkleit der Zugriffsmuster auf unterschiedlichen Zeitskalen
    * Messungen zeigen Selbstähnlichkeit auf Zeitskalen mit 4 Größenordnungen Unterschied (Sekunde - Tag)
  - Charakterisierung mittels
    * Mittelwert,
    * Verhältnis mittlere Ankunftsrate - Spitzenankunftsrate a und
    * Zeitantail, in dem die mittlere Ankunftsrate überschritten wurde b
  - Festlegung der benötigten Kapazität und Bandbreite für Lastspitzen ist schwierig (z.B. schon für $a = 6$ und $b = 5\%$ kann Durchsatz eines Websevers um $12 - 20\%$ kleiner sein als bei homogenen Last)
  - manchmal sind Lastspitzen auf Grund exogener Ereignisse vorhersagbar
  - Berücksichtigung von Zugriffsmustern mit hohen Varianzen ist schwierig, führt zu komplexeren Modellen (einfache Möglichkeit, Vergößerung der Bedienzeiten)

Beobachtung der Verteilung der Bedienzeiten

- Extreme Variabilitäten der Last
  - Dokumentengröße von 10^2 bis 10^6 Bytes
  - Verteilung der Dokumentengröße mit *heavy tail*
    * d.h. sehr große Dateien existieren mit nicht zu vernachlässigender Wahrscheinlichkeit
    * durchschnittliche Werte haben große Varianz und reduzieren Aussagekraft von Mittelwerten
    * mathematische Beschreibung für $heavy\ tails$: $P[X > x] = kx^{-\alpha}L(x)$ wobei $L(x)$ eine sich langsam ändernde Funktion ist
    * Spezialfall (oft verwendet) Pareto-Verteilung mit
      $$P[X < x) = \begin{cases} 1 - \left(\frac{k}{x}\right)^{\alpha} & \text{falls } x \geq k\ (\alpha, k > 0) \\ 0 & \text{sonst} \end{cases}$$
      und damit $P[X > x] = \left(\frac{k}{x}\right)^{\alpha}$ falls $x \geq 0$ und 1 sonst, damit gilt auch
      $\log(P[X > x]) = -\alpha \cdot (\log x + \log k)$
      lineares Verhalten bei Betrachtung von $\log x$, charakteristisch für $heavy\ tails$
  - Möglichkeit der Verkleinerung der Varianz: Definition von Klassen
    * kleinere Varianz innerhalb einer Klasse aber komplexere Modelle

Extreme Unterschiede in der Zugriffsfrequenz auf einzelne Seiten, Zugriffsfrequenz auf eine Seite gehorcht

Gesetz von Zipf

ursprünglich (1949) entdeckt bei der Analyse der Frequenz des Vorkommens von Worten in Texten

- Seiten werden nach Beliebtheit sortiert, dann ist die Zugriffshäufigkeit $f(i)$ der i-ten Seite proportional zu
  $$f(i) \sim 1/i$$
  d.h. auf i-tes Dokument wird doppelt so oft wie auf $2i$-te Dokument zugegriffen

- Verallgemeinerte Form von Zipfs Gesetz:
  $f(i) \sim C/i^{\alpha}$ mit $C, \alpha > 0$

- Zugriffshäufigkeiten von Seiten ändert sich relativ schnell (beliebte Seiten werden unbeliebt, neue Seiten mit häufigem Zugriff tauchen auf)

- Beschreibung der Zugriffshäufigkeit kann für
  - Spiegelung und
  - Caching von Dokumenten ausgenutzt werden
  - möglichst "große" Seiten mit "häufigem Zugriff" im Cache speichern und spiegeln
  - aber Caching verändert Zugriffsfrequenz bei allen nachfolgenden Servern!

2.5 Ein Ansatz zur Kapazitätsplanung von Client-Server Systemen

Planung der Kapazität von C/S-Systemen muss systematischer Vorgang sein

Ziel:

Kostengünstige Realisierung eines adäquaten Systems!

Schritte der Kapazitätsplanung

- Verstehen des Systems bestehend aus HW/SW Umgebung
- Modellierung des Benutzerverhaltens
- Charakterisierung der Last
- Erstellen, validieren und kalibrieren eines Lastmodells
- Erstellen, validieren und kalibrieren eines Systemmodells
- Vorhersage zukünftiger Last
- Analyse der jetzigen und Vorhersage der zukünftigen Systemleistung
- Entwicklung eines Kostenmodells
- Vorhersage der Kosten
- Kosten-/Leistungsanalyse

Dieses Vorgehen umfasst drei Modelle:

- Lastmodell
  - Typisierung von Lastklassen
  - Ressourcenbelastung durch einzelne Klassen
  - Ankunftsfrequenz der Anfragen

- Leistungsmodell
  - Beschreibung der Zuordnung von Last zu (virtuellen) Maschinen
  - Darstellung als (analytisch) analysierbares Modell
  - Berechnung von Antwortzeiten, Durchsätzen und Auslastungen als Funktion der Last und Maschine

- Kostenmodell
  - Bestimmung der Kosten für Software, Hardware und Kommunikation
  - Darstellung der Relation Kosten zu Leistung

Probleme und Fehler bei der Kapazitätsplanung

- Keine einheitliche Terminologie
  - Vielzahl von Werkzeugen und Ansätzen
- Keine einheitliche Definition des Begriffs Kapazität
  - uneinheitliche Definition von Leistungsmaßen
  - unterschiedliche Kapazitäten für ein System
    z.B. nominale Kapazität, nutzbare Kapazität
- Keine Standard-Lasteinheit
- Zukünftige Anwendungen sind schwer vorherzusagen
  - Technologiesprünge sind kaum zu berücksichtigen
  - Vorhersagen für einen Systemtyp verlieren Gültigkeit für anderen Typ
    (z.B. Mainframe - Client/Server)
- Keine Einheitlichkeit zwischen verschiedenen Anbietern
  - unterschiedliche Interpretation der Leistungsangaben
- Messung in komplexen Systemen ist schwierig
  - verkompliziert auch die Validierung

Zentrale Frage: Was heißt adäquate Kapazität?

Typische Beispiele:

- Antwortzeit für eine einfache DB Anfrage soll 2 Sekunden nicht überschreiten
- Antwortzeit für eine DB Abfrage durch die Web-Schnittstelle soll 3 Sekunden nicht überschreiten
- Der NFS Server soll mindestens 10000 E/A-Operationen pro Sekunde mit einer mittleren Antwortzeit von weniger als 0.1 Sekunden erreichen können, die Verfügbarkeit des C/S-Systems soll so groß sein wie in einer vergleichbaren Mainframe-Umgebung
- Web Server soll 99% Verfügbarkeit und weniger als 1 Sekunde Antwortzeit für 90% der Anfragen garantieren

Auch bei diesen Festlegungen sind noch Begriffe genau zu spezifizieren (z.B. Was ist eine "einfache DB Anfrage"?)

$\Rightarrow$ *Service-level agreements (SLAs)* sagen aus,
was der Benutzer vom System erwarten darf

Spezifikation von SLAs durch Benutzer und Betreiber

SLA ist ein Vertrag zwischen Benutzer und Betreiber
(u.U. auch mehrere Betreiber)

Typische Struktur eines SLA

- *Hintergrund*: Beschreibung des Geschäftsumfeldes
- *Beteiligte*: Kontaktpersonen und Verantwortliche in den beteiligten Gruppen und Abteilungen
- *Dienstdefinition*: Definition des zu erbringenden Dienstes inkl. Ort, wo der Dienst erbracht werden muss, der Benutzerzahl, des Zeitfensters
- *Voraussetzungen*: Infrastruktur des Benutzers
- *Leistungsmaße*: Aufzählung der geforderten Leistung bei vorgegebener Last (genaue Spezifikation!!)
- *Messmethode*: Definition einer Methode, mit der Leistung gemessen werden soll
- *Problembericht*: Definition eines Formats und eines Adressaten für Problemberichte
- *Konsequenzen*: Festlegung der Konsequenzen, falls angegebene Leistungsmaße nicht erreicht wurden (Konventionalstrafen, Kostenübernahme)
- *Konfliktlösung*: Definition einer Konfliktlösungsstrategie
- *Änderungsintervalle*: Intervalle in denen das SLA überarbeitet werden muss

Spezifikation von benutzbaren Technologien und Standards

- Homogene Umgebung (nur XP oder nur Unix)
- Standardprotokolle
- Aspekt der Administration und Wartbarkeit

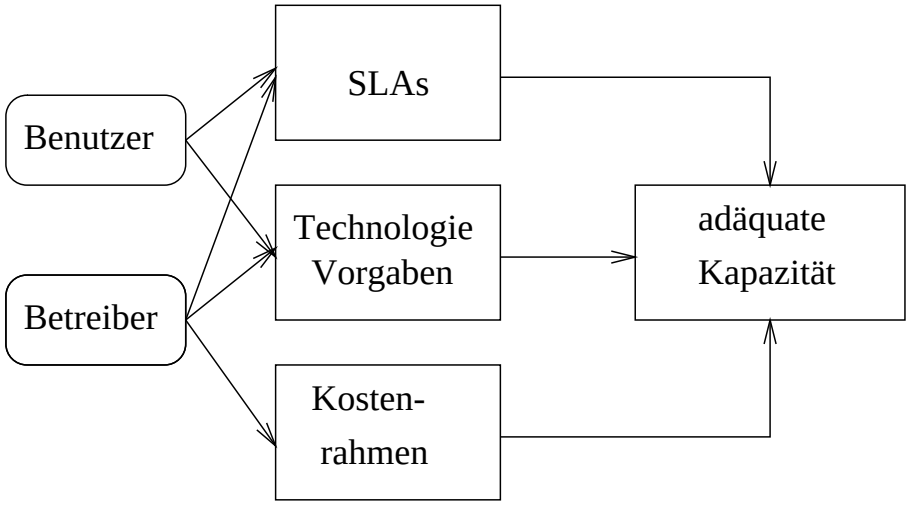
Kostenrahmen

- Beschränktes Budget vorhanden
- Kosten in Investitionskosten und laufende Kosten unterteilen
- Kosten in Relation zu Leistung oder Erlösen setzen

Eine Definition

Ein C/S-System hat eine adäquate Kapazität, wenn die SLAs erfüllt werden, die technologischen Randbedingungen eingehalten werden und der Kostenrahmen nicht überschritten wird.

Darstellung der Situation



Typische Anforderungen

- Mittlere Antwortzeit kleiner 2 Sekunden
- NT Server mit TCP/IP
- Investitionskosten < 1 Mio. Euro
- laufende Kosten < 50000 Euro pro Jahr

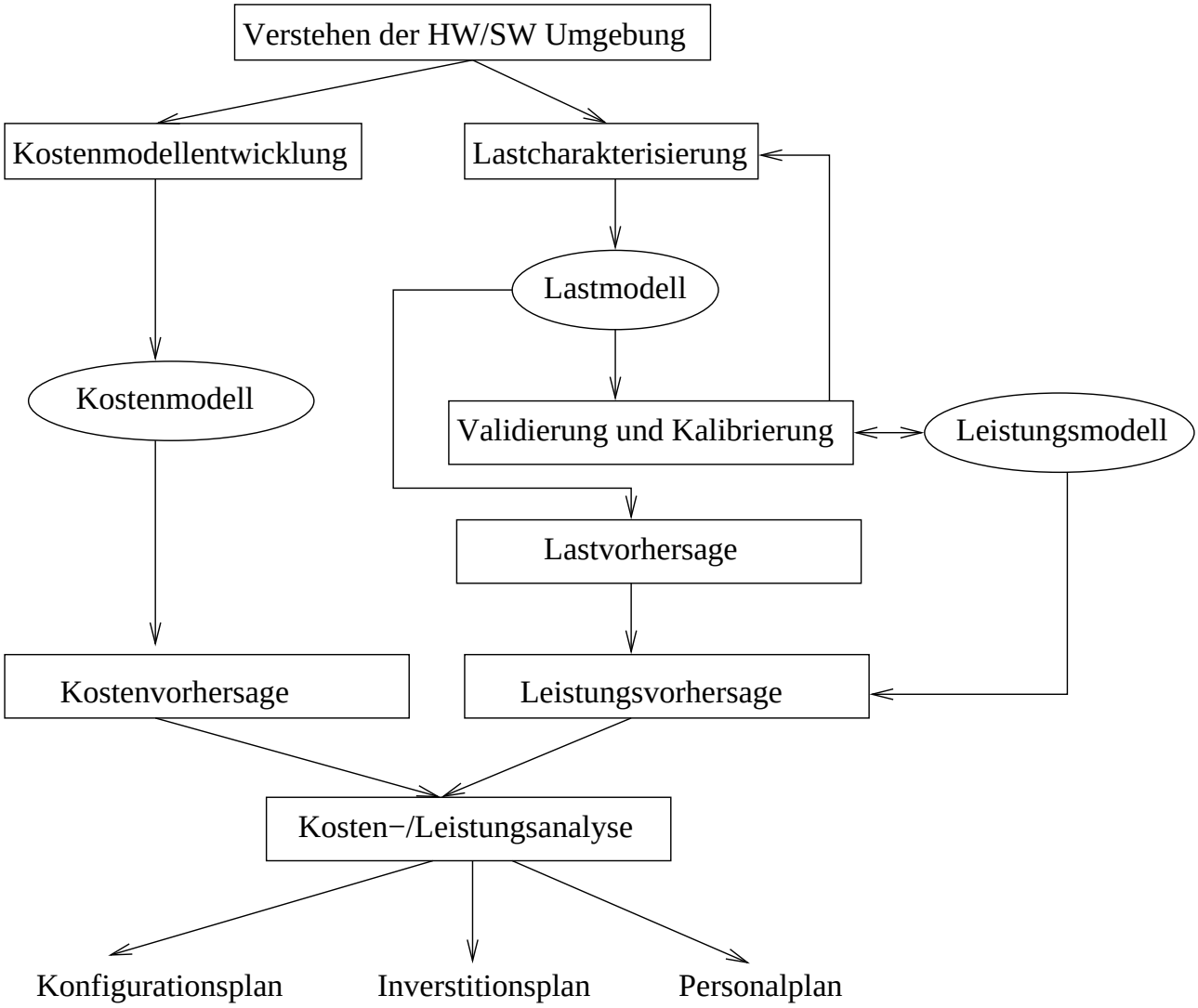
Konfiguration umfasst neben HW/SW auch

- Realisierung der Vernetzung
- Lage der Server

U.U. existieren mehrere Konfigurationen,
die adäquate Kapazität bereitstellen, dann Auswahl nach Kosten-/Leistungsgesichtspunkten

2.5.1 Ein Vorgehensmodell

Hier betrachtet:
Vorgehensmodell nach Mensace,
andere Modelle existieren

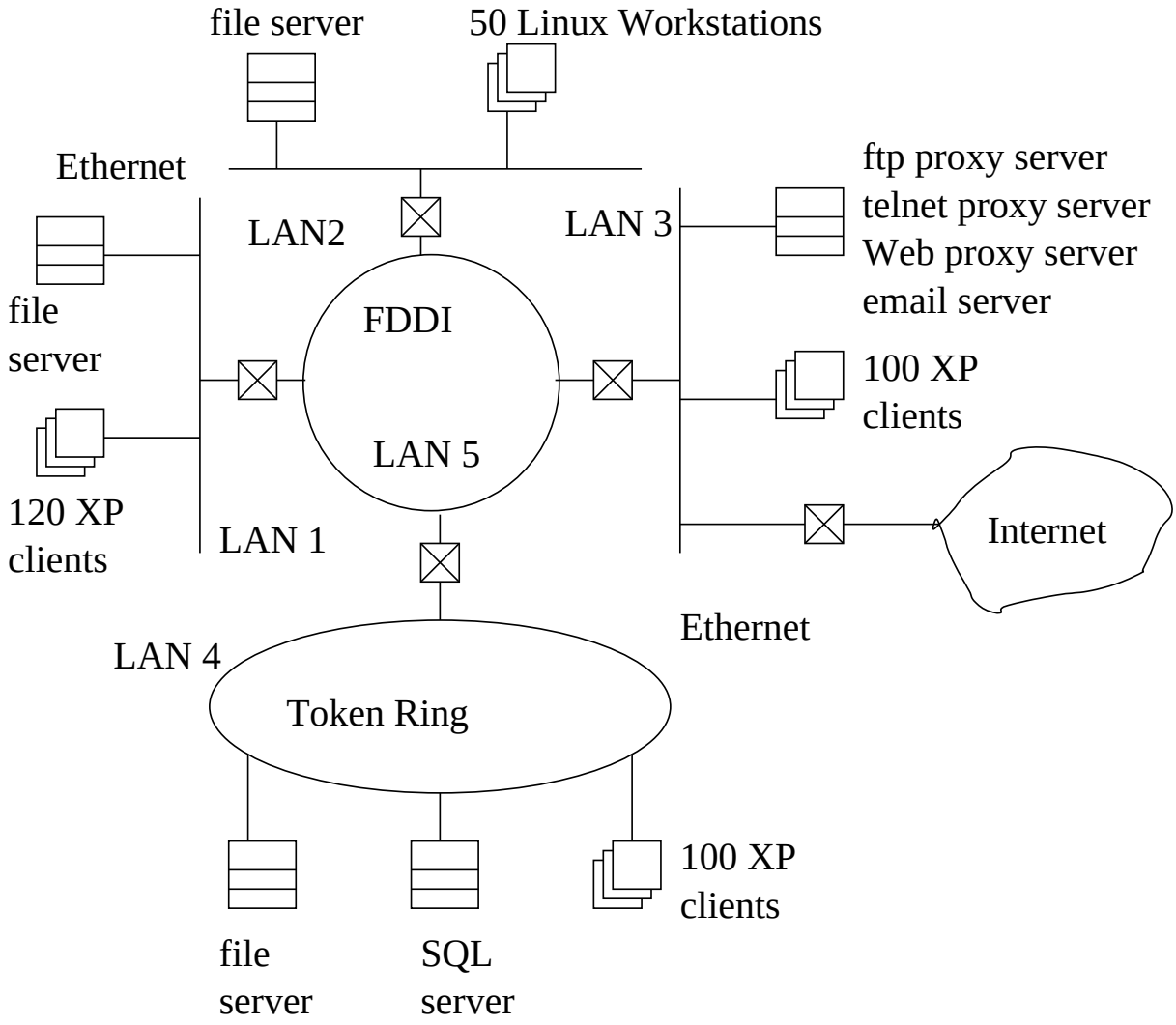


Verständnis der Umgebung

Elemente, die verstanden werden müssen

Element	Beschreibung
Client Plattform	Anzahl und Typ
Server Plattform	Anzahl, Typ, Konfiguration
Middleware	Typ (z.B. TP Monitor)
DBMS	Typ
Anwendungen	hauptsächliche Anwendungen
Netzwerkverbindungen	Verbindungsdiagramme für alle LANs und WANs, Technologieder Netze, Router und Server Clients per LAN
Netzwerkprotokolle	Liste der verwendeten Protokolle
SLAs	vorhandene SLAs oder Industriestandards
LAN Management	Unterstützung beim LAN Management, Erfahrung
Beschaffungsprozedur	Verantwortlichkeiten, Dauer, Ausgabenlimits

Eine Beispielkonfiguration



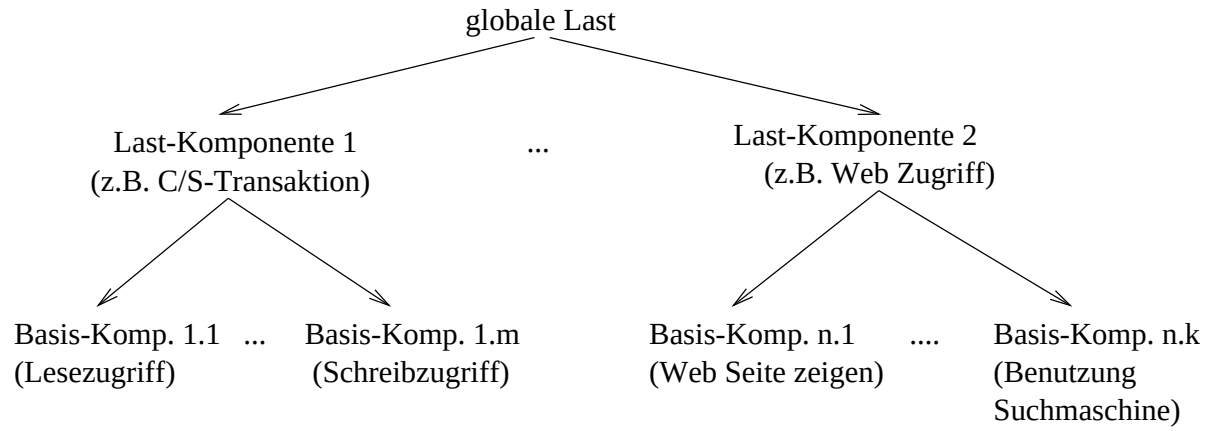
Folgende Schritte:

- Lastcharakterisierung
- Leistungsmodellerstellung
  (Was ist wie detailliert zu modellieren?)

2.5.2 Charakterisierung der Last

Lastcharakterisierung ist der Prozess der präzisen Beschreibung der globalen Systemlast in Form von primären Lastkomponenten

Jede Lastkomponente wird anschließend in Basiskomponenten dekomponiert



Die Basiskomponenten sind charakterisiert durch

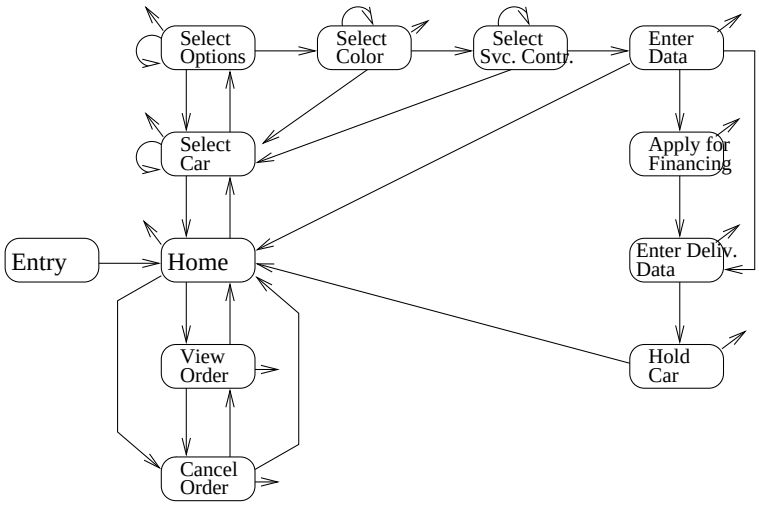
- Lastintensitäten (d.h. Ankunftsraten) und
- Bedienanforderungen für die Systemressourcen

Lastparameter können oft nicht direkt aus Messungen gewonnen werden, sondern müssen aus gemessenen Werten abgeleitet werden
(d.h. insbesondere aus dem Benutzerverhalten) !

Nachdem Last charakterisiert und gemessen
erfolgt Aufbau des Lastmodells

Abbildung des BVGs auf primäre Lastkomponenten (über BBM und CSID)

Vorgehen am Beispiel eines Internet-Gebrauchtwagenhandels



System ist auf Ebene der Benutzermodellierung beschrieben

Realisierung der Funktionen des BVGs durch vorhandene IT-Infrastruktur

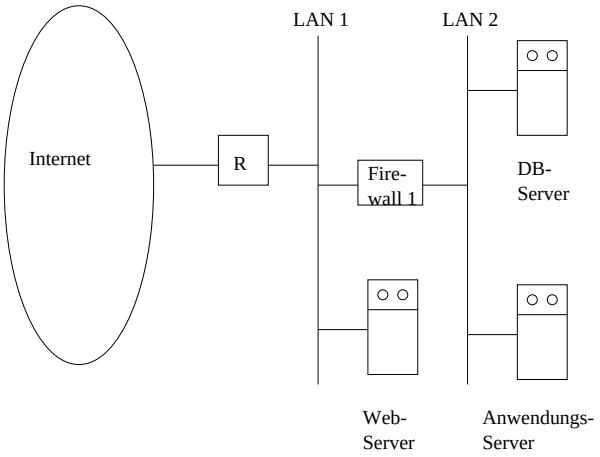
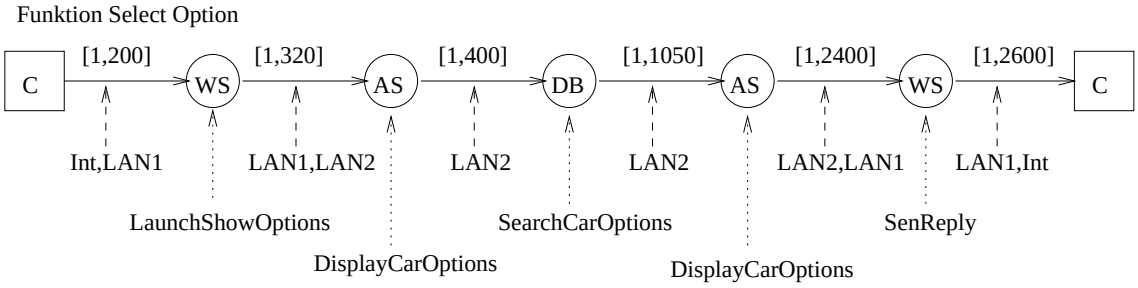


Abbildung der einzelnen Funktionen des BVGs auf angebotene Funktionen des C/S-Systems durch CSID pro Funktion



Bisherige Beschreibung zeigt funktionalen Zusammenhang
durch quantitative Information über BVG und Hardware
entsteht Lastmodell

Einzelne Funktion im CSID kann verschiedene Komponenten der jeweiligen Hardware belasten
(je nach Detaillierungsgrad der Modellierung)
z.B. Belastung von CPU und Platte

Beschreibung der Ressourcenbelastung
durch Angabe der mittleren Zeitdauern

Am Beispiel: Belastung durch Funktion $SelectOption$:

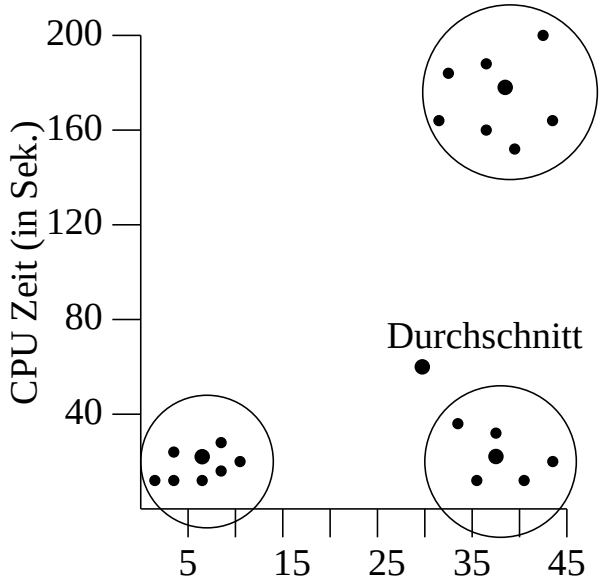
Server			LAN 1	LAN 2	In.-
WS	AS	DB			Verbind.
C: 4.8 P: 8.5	C: 18.0 P: 14.0	C: 13.0 P: 20.0	0.328	0.352	12.0

Belastung in msek: C = CPU, P = Platte

Unterscheidung in Lastklassen

Beispiel C/S-System während einer Stunde beobachtet bzgl. CPU Zeit und Anzahl E/A-Operationen

Ergebnis graphisch repräsentiert



Naheliegend Darstellung durch 3 Lastklassen mit den Mittelwerten (4.5,19),(38,171),(39,22)

Offensichtlich realistsicher als Darstellung durch Mittelwert (28,68)

In Realität Darstellung nicht so einfach:

Algorithmen zur Cluster-Bestimmung existieren
($\Rightarrow$ später mehr)

Also funktional- + verhaltens-bestimmte Lastklassen!

Datenermittlung

Ideale Situation: alle Daten werden genau erhoben

Probleme:

- schlechte Messbarkeit
- fehlende Messmöglichkeiten
- fehlende Zeit
- Möglichkeiten des Vorgehens:

Messung eigentlich für jeden Client und jeden Server

- oft Kompromisse nur einige Messungen und
- Interpolation der Ergebnisse für ähnliche Systeme
  oft verwendet SPEC oder TPC Benchmarks

Also Berechnung der Form
Bedienanforderung =
  gemessener Wert * Durchsatz Verhältnis
(später mehr zu Messung und Benchmarking)

Validierung des Lastmodells

Jedes Modell enthält Abstraktion

Validierung muss sicherstellen, dass Modell reales
Verhalten (bzgl. gewünschter Resultate)
nur im vertretbaren Maße verfälscht wird
(hier Verfälschungen von ca. 10%-30% tolerabel)

Falls Validität nicht vorhanden $\Rightarrow$
Kalibrierung des Modell (d.h. Anpassung an Realität)

Vorsicht vor Überanpassung

(weitere Details Vorlesung Modellierung und Simulation)

Lastvorhersage

Ziel Vorhersage der Last in X Zeiteinheiten

Vorgehen:

- Betrachtung bisheriger Entwicklungen
- Einbeziehung zukünftiger Randbedingungen
- Aufstellung eines mathematischen Modells
  zur Vorhersage
  (z.B. lineares Regressionsmodell, moving average,
  exponential smoothing)

$\Rightarrow$ später mehr zur Lastvorhersage

2.5.3 Leistungsanalyse

Ziel der Leistungsanalyse:

Ermittlung der Leistungsgrößen aus der Last- und Systembeschreibung

Eingabeparameter:

- Systemparameter (z.B. Netzwerkprotokoll, Datenverteilung, maximale Anzahl von Threads im Server)
- Ressourcenparameter (z.B. Suchzeiten von Platten, Bandbreite des Netzes, CPU Geschwindigkeit)
- Lastparameter
  - Ankunftsraten
  - Bedienbedarf

Verwendete Modelltypen:

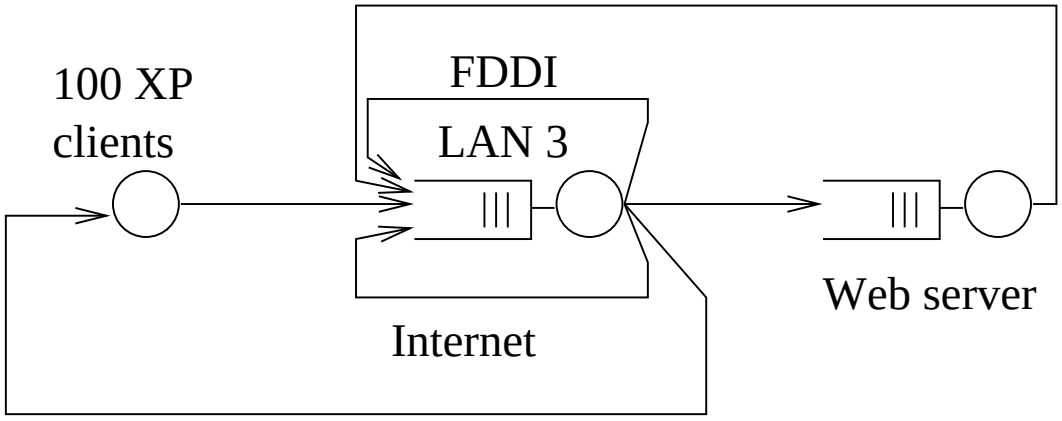
- Simulationsmodelle
- analytische Modelle

in beiden Fällen auf Warteschlangennetzen basierend

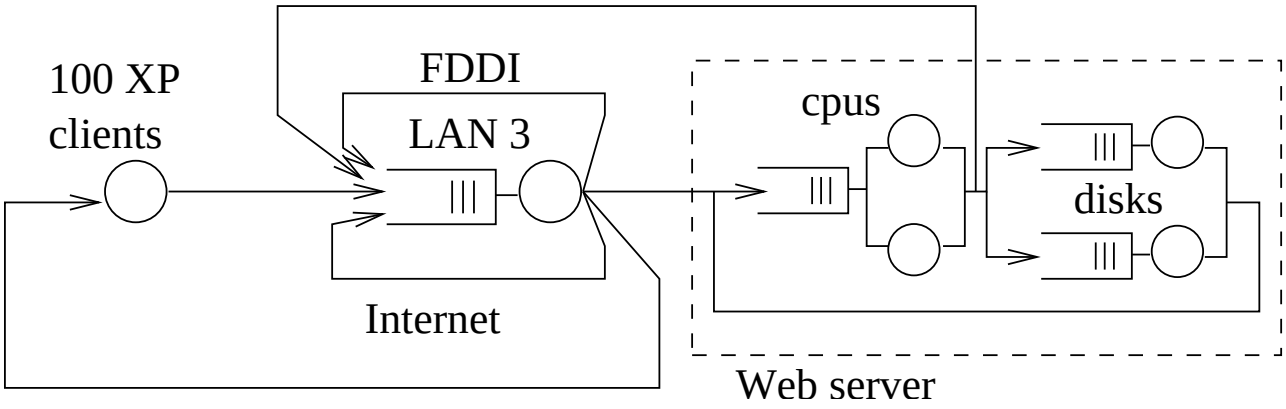
Warteschlangen modellieren

- Hardware-Ressourcen (CPU, Platte, Router)
- Software-Ressourcen (Threads, DB Locks)

Beispiel (LAN 3 auf Folie 102)



Verfeinerte Darstellung des Web Servers



$\Rightarrow$ später mehr zur Analyse der WS-Netze

Techniken der Leistungsbewertung

Ziel insgesamt Lösung/Analyse des spezifizierten Leistungsmodells

- Analytische Lösung
  - Menge von Formeln müssen gelöst werden
  - Analysealgorithmen existieren
  - Effizientes Vorgehen
  - Exakte Ergebnisse
  - Beschränkung auf abstrakte Beschreibung

- Simulative Lösung
  - Nachspielen des dynamischen Verhaltens
  - Statistische Ergebnisse
  - Aufwendige Analyse
  - Prinzipiell kaum Beschränkungen bei der Modellkomplexität

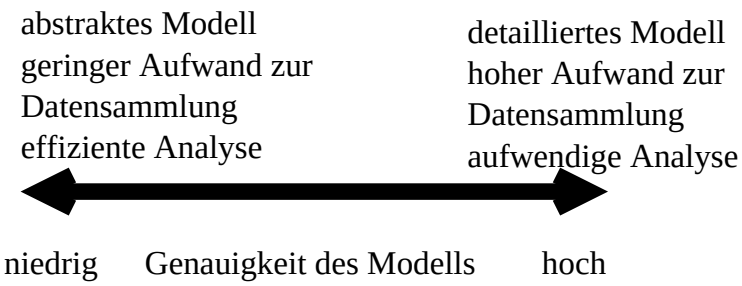
Beachten:

Detailliertere Modelle erfordern detailliertere Daten!

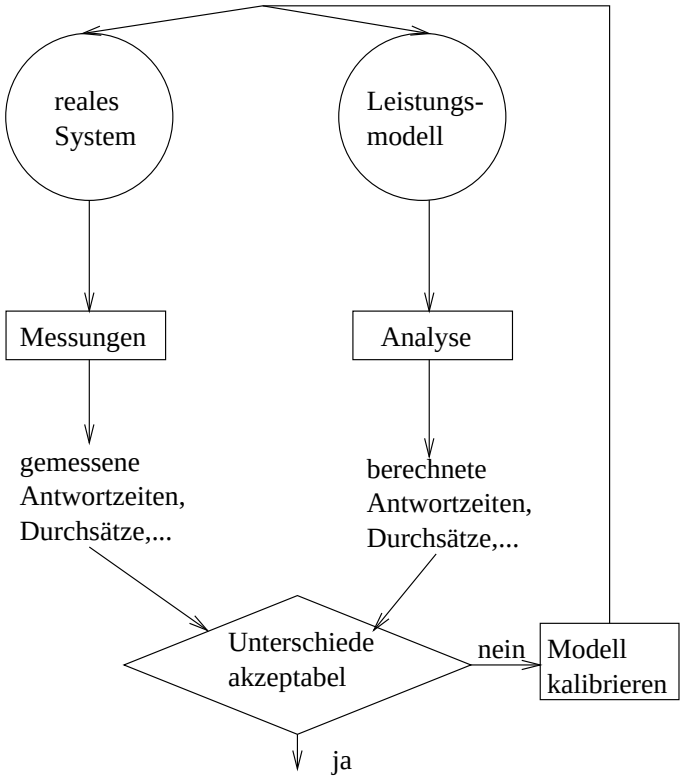
$\Rightarrow$ hier nur analytische Modelle

Validierung des Leistungsmodells

Genauigkeit des Leistungsmodells



Vorgehen bei der Validierung



Für die Kapazitätsplanung ist ein Fehler von 10%-30% akzeptabel

2.5.4 Kostenanalyse

Kapazitätsplanung muss Kostengesichtspunkt beachten

Kosten müssen möglichst detailliert auf
ihre Verursacher zurückgeführt werden

Man unterscheidet

- Investitionskosten
- laufende Kosten

Größe und Heterogenität von C/S-Systemen erschweren
Zuordnung von Kosten zu Verursachern

Kosten für C/S-Systeme können unterteilt werden in

- Kosten für Hardware: Rechner, Speichermedien, Router, Verkabelung, Wartung
- Kosten für Software: BS, DBMS, Büroautomatisierung, Anwendungen
- Kosten für Kommunikationssysteme: WAN Anbindung, Standleitungen, Internet Service Provider
- Kosten für Betriebsunterstützung: Systemadministratoren, *Help desk* Unterstützung für Benutzer, Ausbildungskosten, Kosten Netzwerkmanagement und -monitoring Software

Erhebungen zeigen:

- 60%-70% der Kosten sind Personalkosten
- 10% der ursprünglichen Investitionskosten pro Jahr für Erneuerungen von Soft- und Hardware
- Administrationskosten 500-700 pro Jahr und Rechner (!?)
- Ausbildungskosten 1500-3500 pro Mitarbeiter

Wenn Kosten- und Leistungsmodell vorhanden,
kann kombinierte Kosten-/Leistungsanalyse erfolgen

Typische Fragestellungen:

- Was kostet die Halbierung der Antwortzeit bei Web Zugriffen?
- Welche Leistungssteigerung und Kosten folgen aus einer Umstellung von einer zwei- auf eine dreistufige C/S-Architektur?
- ....

Resultat der Analyse:

- Konfigurationsplan
- Investitionsplan
- Personalplan