veritas
iustitia
libertas
Freie Universität Berlin

Ansätze zum Schließen der semantischen Lücke zwischen Anwendung und Betriebssystem

Thiemo Voigt, Hartmut Ritter
FU Berlin, AG Technische Informatik

Überblick

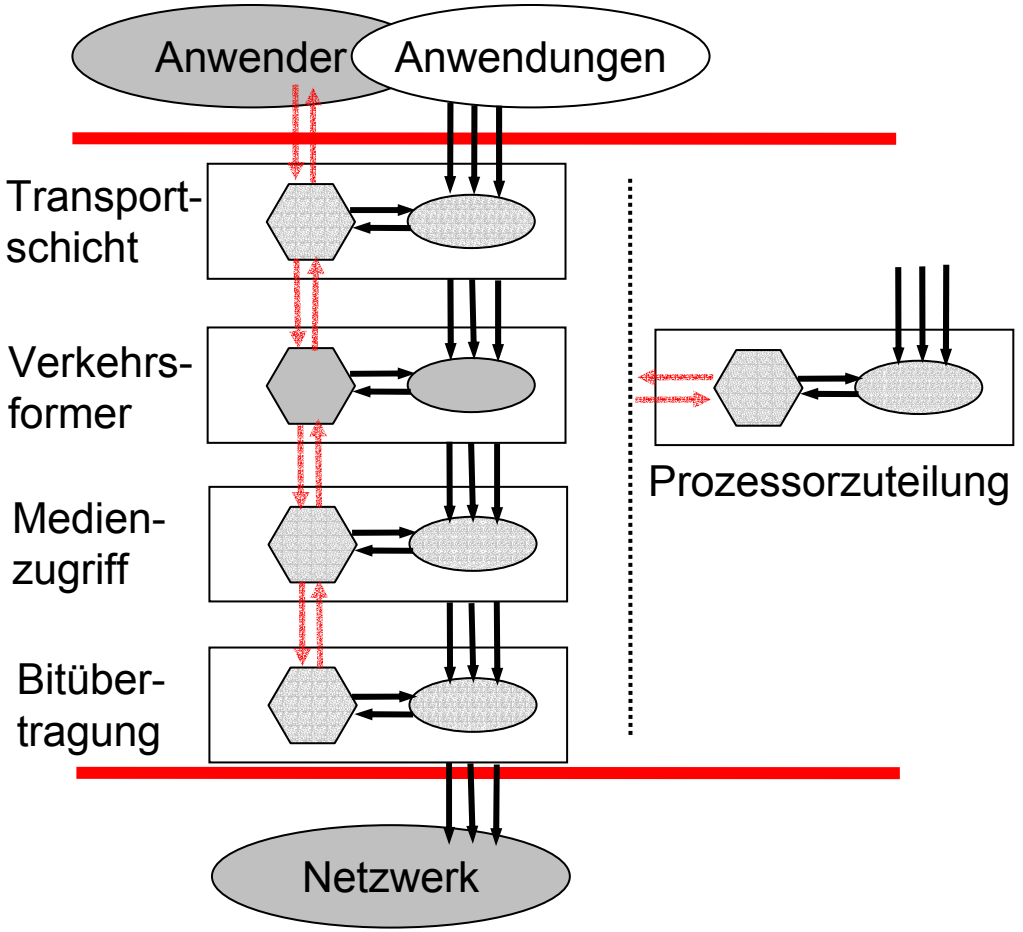
- Semantische Lücke zwischen Anwendung und Betriebssystem
- Dienstgüte bei “klassischen” Anwendungen
- Dienstgüte und virtuelle Maschinen
  - Größere Lücke?
- Anwendungsbedeutung und Dienstgüte
  - Ausblick: verteilte Spieleplattformen

Semantische Lücke

- Lücke zwischen:
  Dienstgüteanforderungen der Anwendung / des Anwenders
  und
  technischen Parametern des Betriebssystems

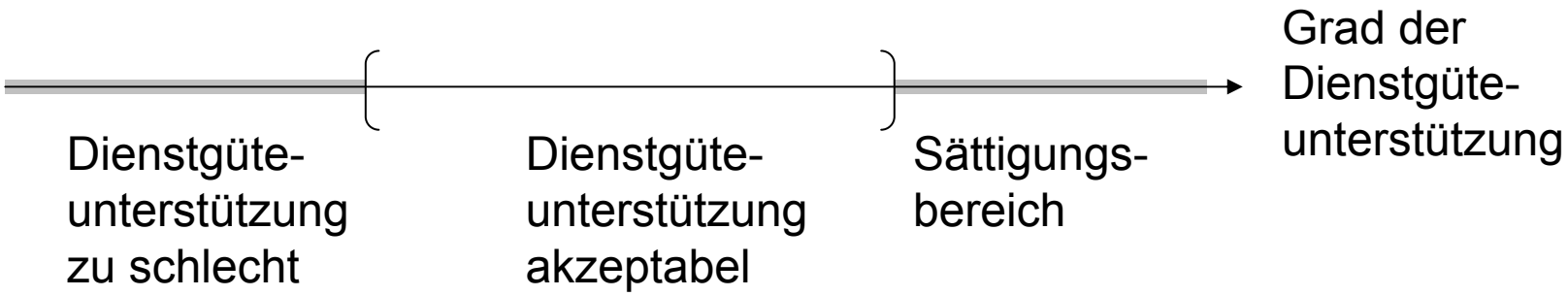
- “Klassische Dienstgüteparameter”:
  Datenrate, Jitter, Rechenzeit

- Bedeutung von Aktionen für eine Anwendung



Dienstgüte-Begriff

- **Dienstgüte** (ITU E.800):
  Grad der Zufriedenstellung eines Benutzers



- **Adaptivität eines vernetzten Endsystems:**
  Fähigkeit zur selbsttätigen Anpassung der Systemparameter an eine Zielfunktion

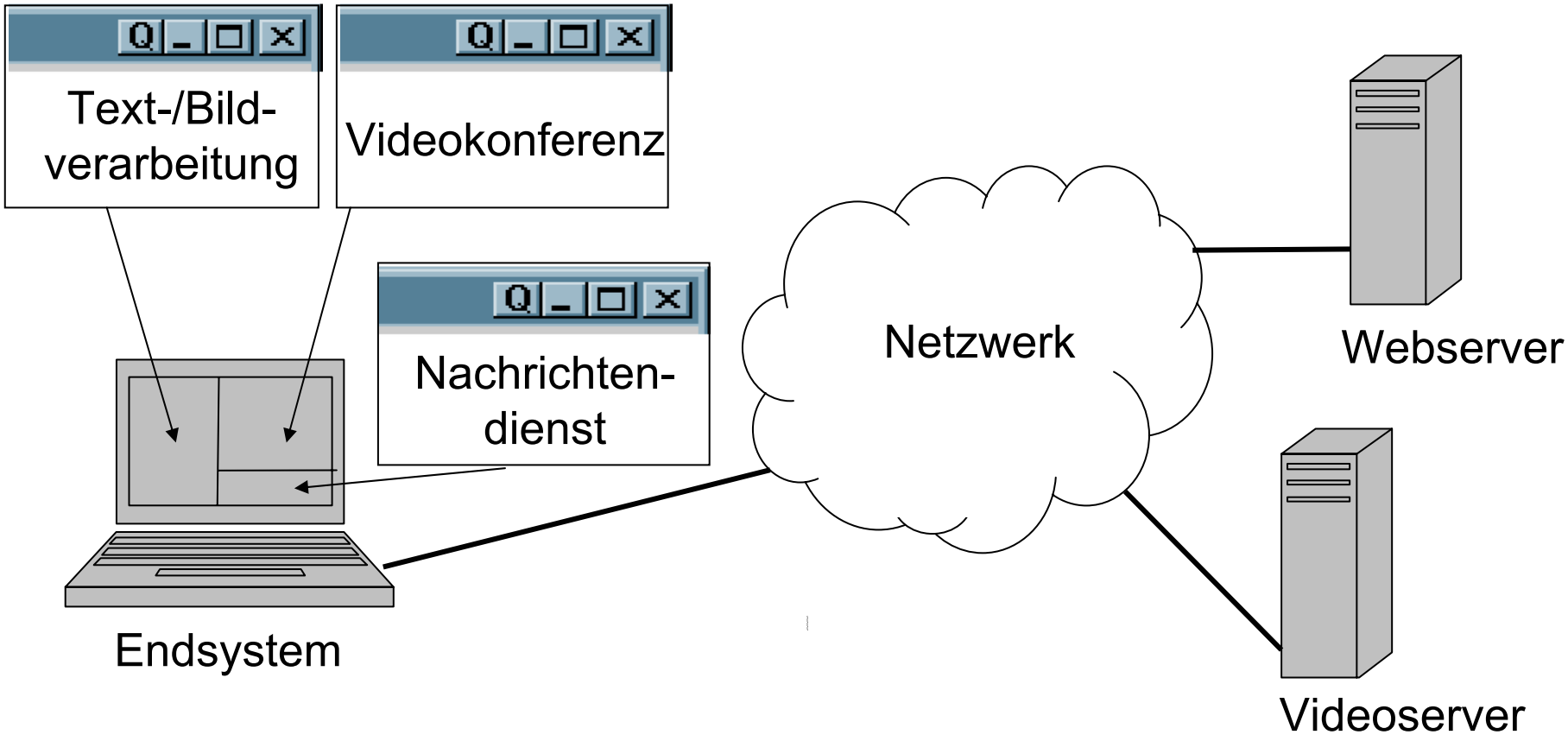
- **Zielfunktion:**
  Grad der Dienstgüteunterstützung des Endsystems

Benutzerinteraktion

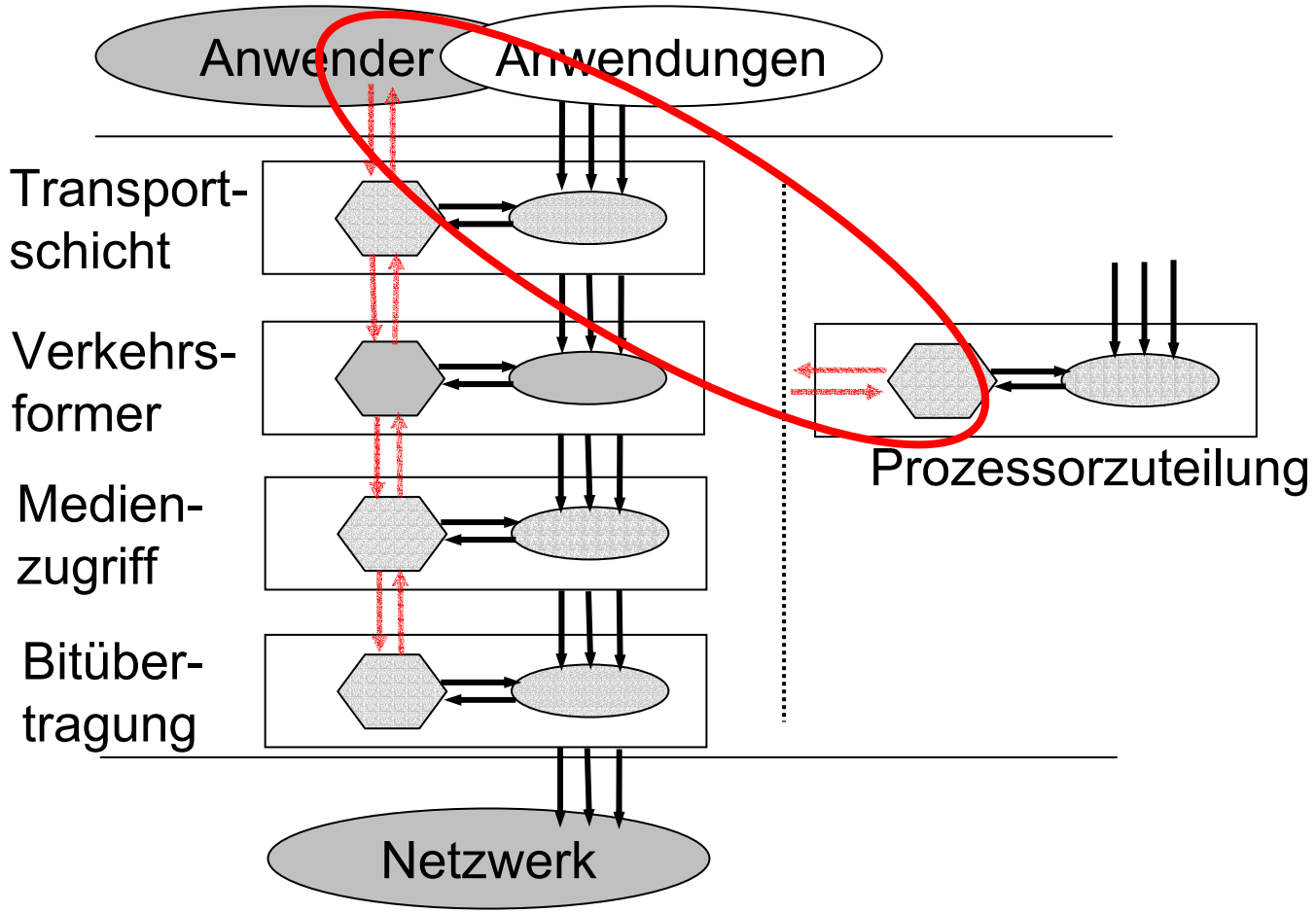
- Neue intuitive Schnittstelle:
  Beeinflussung der Ressourcenverteilung durch den Benutzer
- Realisierung im Fensterverwalter:

Bedarfsorientierung, Beeinflussung der Dienstgüte zur Laufzeit einer Anwendung

Anwendungsbeispiel



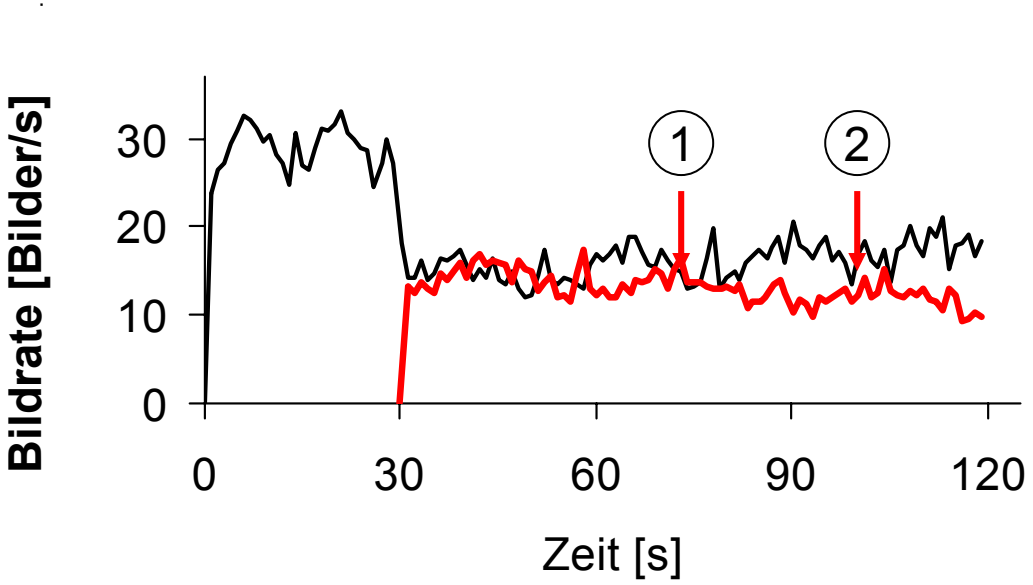
Kopplung Anwendung - Prozessorzuteilung



Anforderungen zur Laufzeit

- Änderung der Rechenzeitzuteilung bei Anwendungen ohne aktive Netzwerkverbindung

- Messung: Bildrate zweier konkurrierender Anwendungen zur Darstellung einer Videosequenz



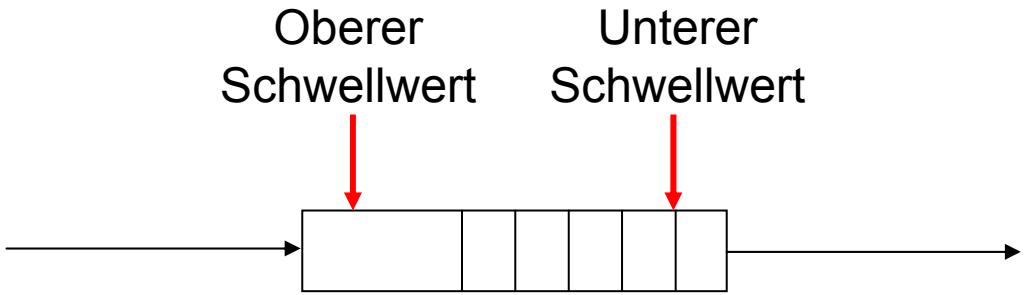
①, ② : Aktivierung der Schaltfläche durch den Benutzer zu Gunsten der zuerst gestarteten Anwendung

Signalbasierte Rückkopplung

- Koordination zwischen Anwendungen und Betriebssystem durch einfache Signale
- Verzicht auf Austausch umfangreicher Parametersätze

⇒ **Skalierbarkeit der Dienstgüteunterstützung**

- Einfache Rückkopplung durch Messung des Pufferzustandes:



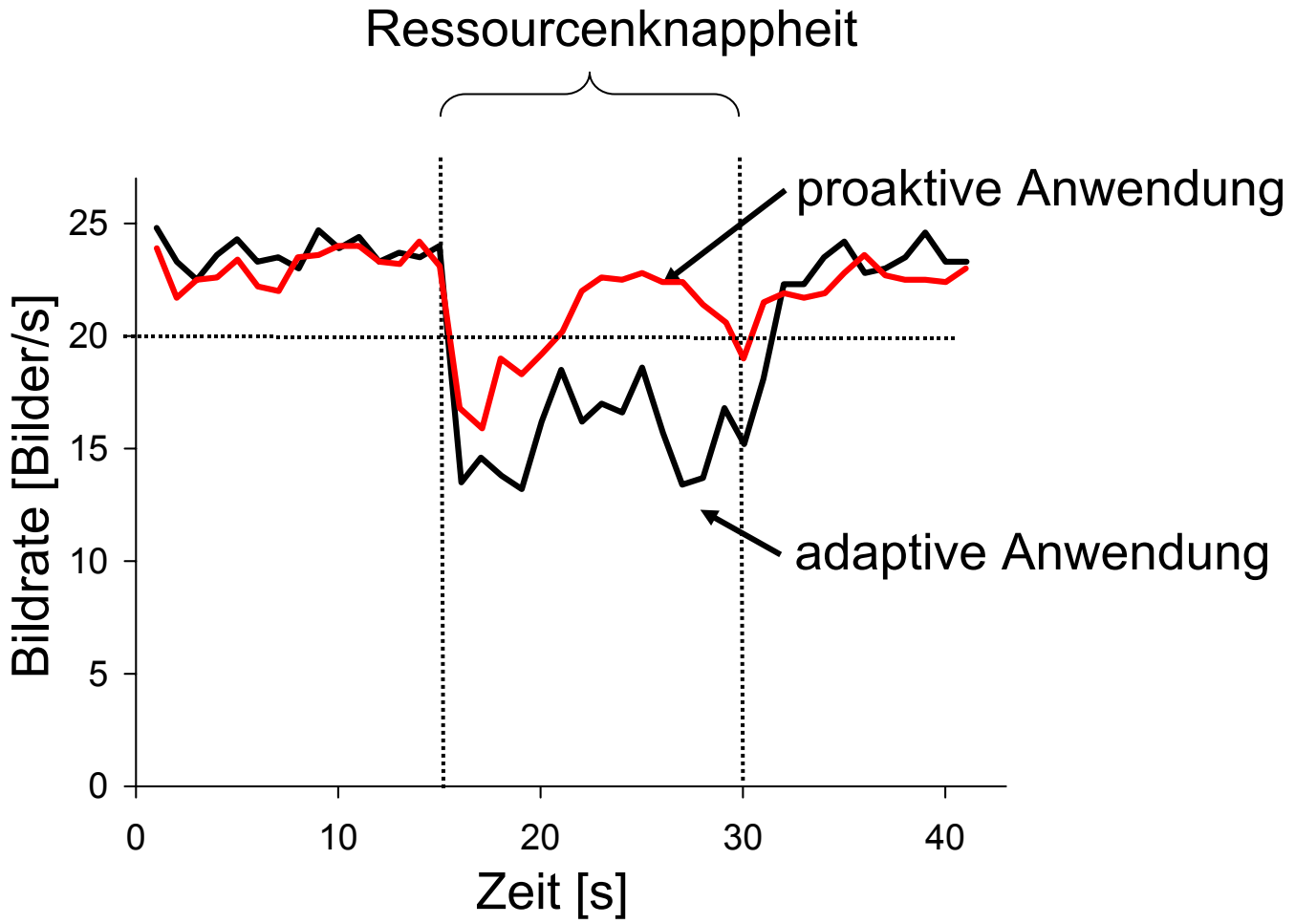
Adaptive und proaktive Anwendungen

- **Adaptive** Anwendung:
  Reaktion auf Verschlechterung der Ressourcenverfügbarkeit durch Anpassungen innerhalb der Anwendung

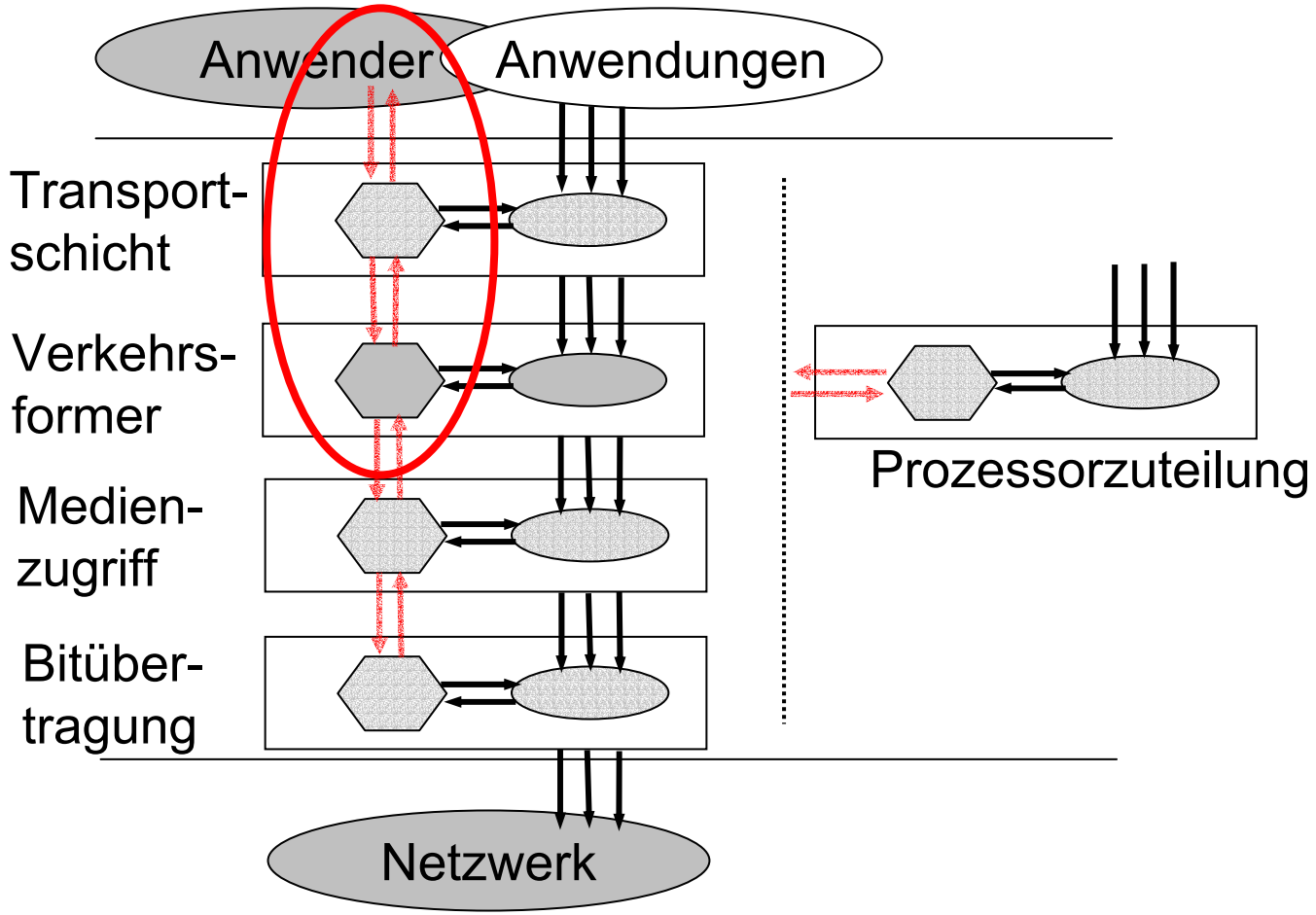
- **Proaktive** Anwendung:
  Nach außen gerichtete Aktion bei Verschlechterung der Ressourcenverfügbarkeit,
  Beeinflussung des umgebenden Systems

- Realisierung durch einfaches Signal der Anwendung,
  auf Grund von Messungen der Ausgangswarteschlange oder anwendungsspezifischer Parameter

Vergleich: Messung unter Linux



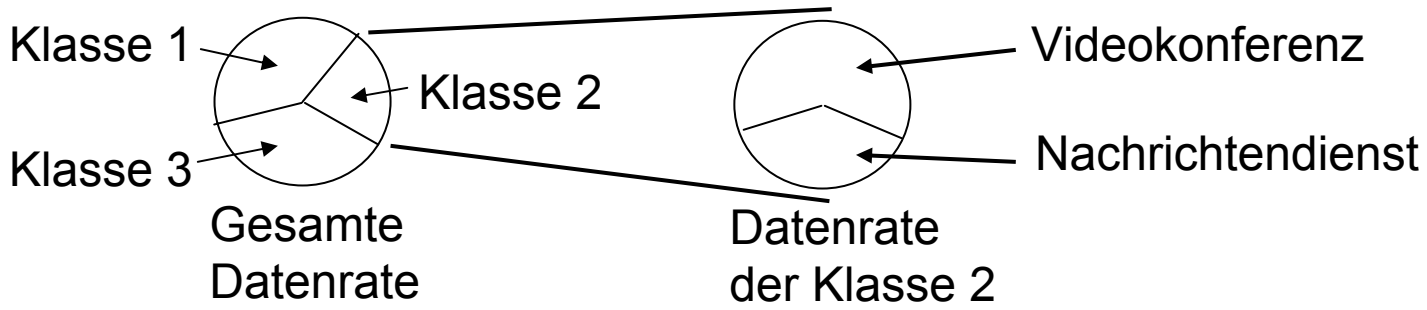
Kopplung Anwendung - Verkehrsformer



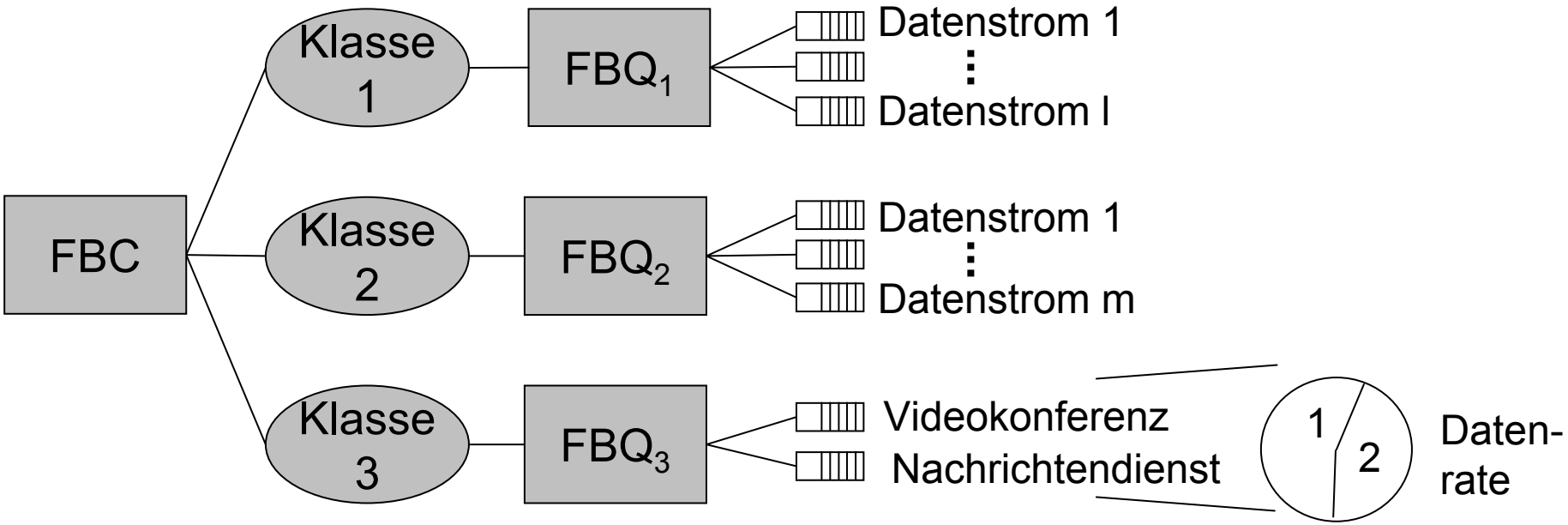
Vermittlungsfunktion des Verkehrsformers

- **Vermittlungsfunktion:**
  Anpassung der Datenrate der Anwendungen an
  - Anforderungen der Anwendungsschicht
  - Begrenzungen des Netzwerks (**Verkehrsvertrag**)

- Anzahl der Klassen und Datenrate je Klasse durch Verkehrsvertrag vorgegeben
- Zuteilung der Datenrate auf Anwendungen innerhalb einer Klasse nach Gewichtung



Aufbau des Verkehrsformers

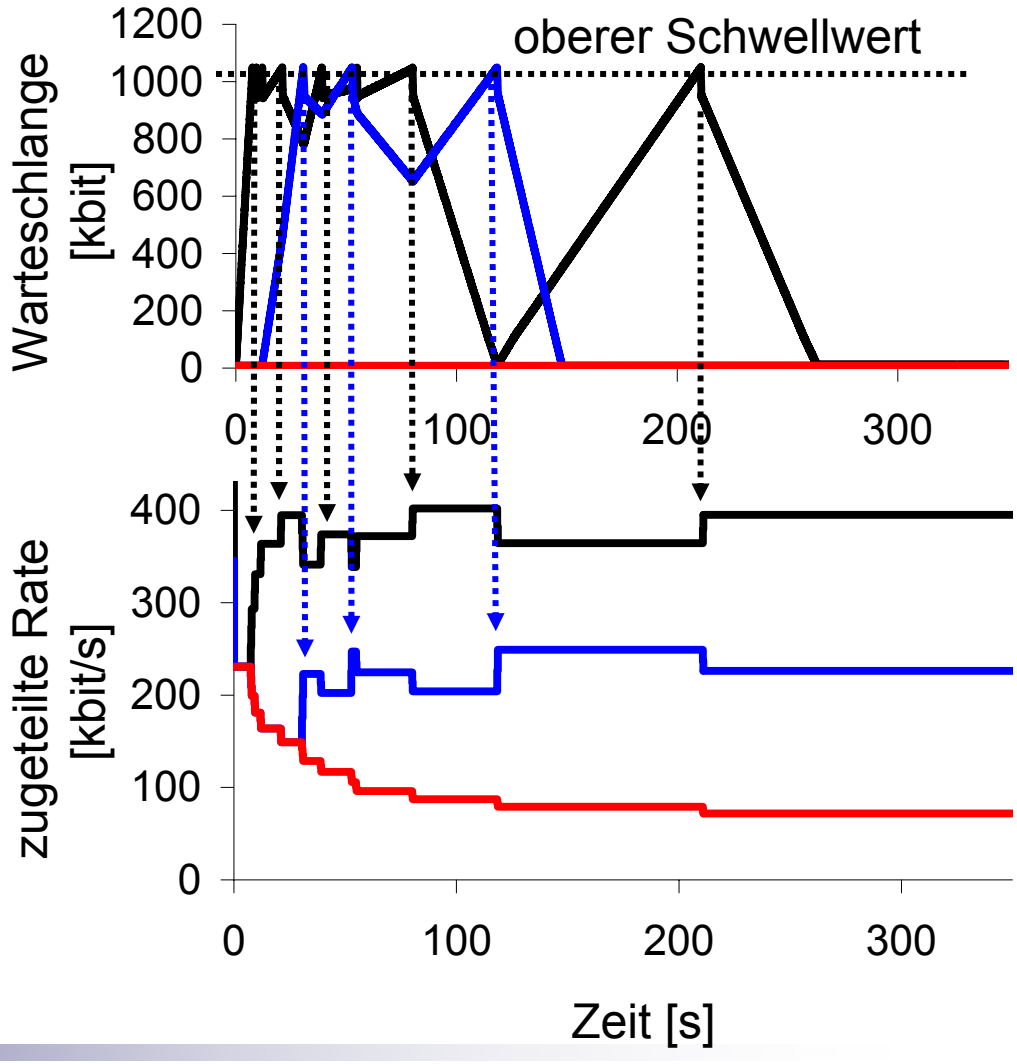


FBC: Klassifizierung eines Datenstroms
FBQ: Wartschlangenverwaltung je Datenstrom

Rückkopplung durch Warteschlangen

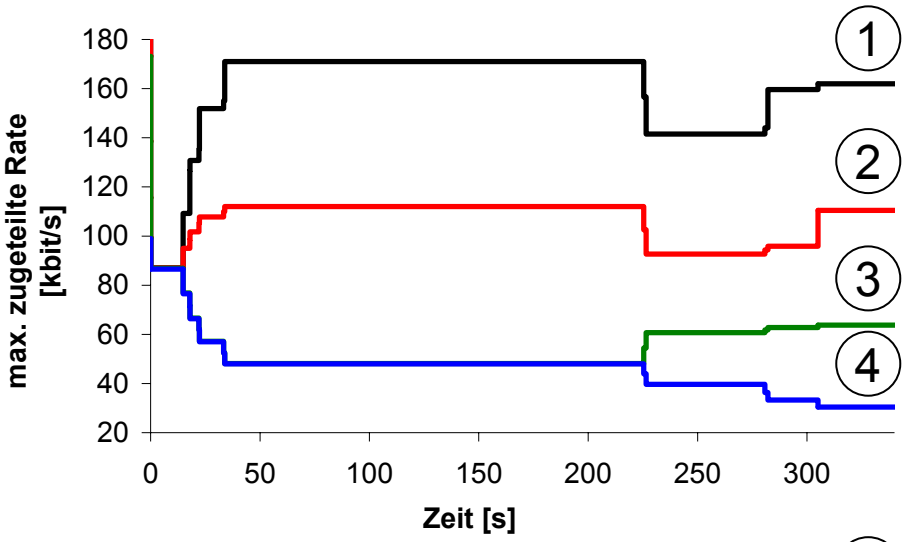
- Selbsttätige Anpassung durch Rückkopplung zwischen proaktiven Anwendungen und Verkehrsformer
- Stabiler Zustand nach Einschwingphase
- Simulation mit OMNeT++

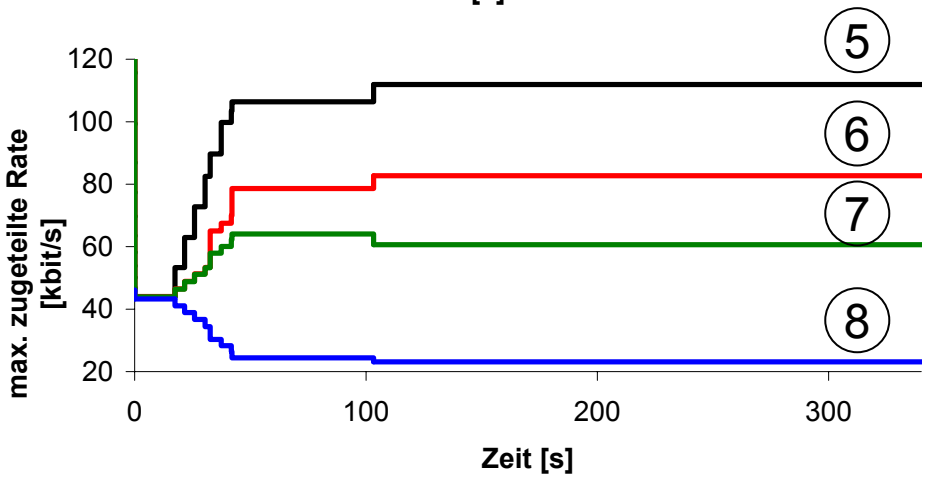
Ratenanforderungen der Anwendungen:
- : 350 kbit/s
- : 200 kbit/s
- : 50 kbit/s



Lösung für viele Datenströme

- 8 bzw. 16 Datenströme
- Erhöhung der Datenrate der anfordernden Verbindung, Reduzierung gering ausgenutzter Verbindungen
- Grad der Erhöhung abhängig von der Gesamtzahl der Verbindungen im System



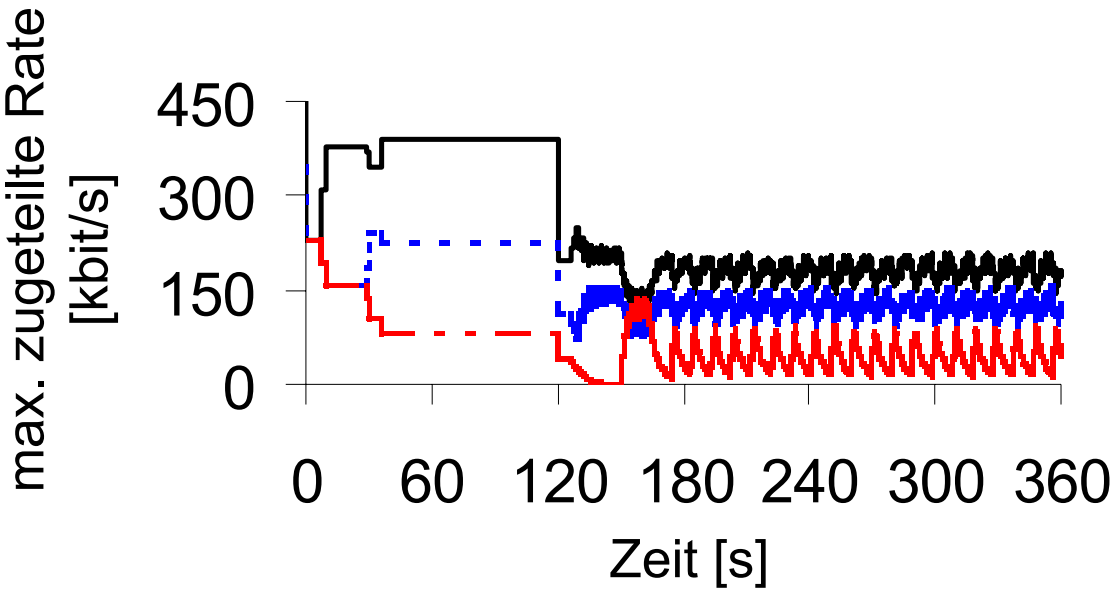


Ratenanforderungen der Anwendungen:

①	: 150 kbit/s	⑤	: 100 kbit/s
②	: 100 kbit/s	⑥	: 75 kbit/s
③	: 50 kbit/s	⑦	: 50 kbit/s
④	: 25 kbit/s	⑧	: 20 kbit/s

Ratenreduktion

- Reduktion der verfügbaren Datenrate nach 120 Sekunden



Ratenanforderungen der Anwendungen:
Anwendung 1: 350 kbit/s
Anwendung 2: 200 kbit/s
Anwendung 3: 50 kbit/s

Ausgangsproblem

- ✓ Abbildung einfacher Anforderungen auf Anpassungsmechanismen im Betriebssystem
  - Prozessorzuteilung
  - Verkehrsformung

- ✓ Einfache Abbildung zwischen Anwendungsfenster und Repräsentation im Betriebssystem

- ? Zuordnung der Anforderungen auf Prozesse zu ungenau, grundverschiedene Anwendungen innerhalb eines Prozesskontexts?

- ? Vergrößerung der Lücke zwischen Anwendung und Betriebssystem durch Zwischenschicht (Java VM, CLR)???

CLR - Common Language Runtime

- virtuelle Maschine, in der alle .NET Anwendungen laufen
- Übersetzung des Quellcode in MSIL (Microsoft Intermediate Language)
- Aufgaben der CLR:
  - Programmausführung
  - Garbage collection
  - JIT-Kompilierung
  - Fehlerbehandlung mit Exceptions

CLR

- Übersetzen des Quellcodes in MSIL code
- Integration in Teil der .exe-Datei (ansonsten win32 portable-executable-Standardformat)
- Import der Funktion _CorExeMain aus der CLR beim Übersetzen/Linken

- Start der Anwendung: Laden der .exe-Datei durch das Betriebssystem, plus alle notwendigen DLLs (dynamic link libraries)
- Importierte Funktion _CorExeMain führt MSIL code aus,
  erste Benutzung: just-in-time (JIT) compilation

Anwendungsprozesse und CLR

- Bedeutung für Ausgangsproblem
  - Ausführung der CLR als Teil des Anwendungsprozesses
  - Keine „background threads“, auch nicht zur Garbage Collection oder zur JIT-compilation

⇒ Vorgestellter Ansatz auch im .NET-Rahmenwerk verwendbar!

Ausgangsproblem

- Gelöst???

- Noch nicht!
  - Gut geeignet für streaming, langdauernde Anwendungen, weniger geeignet für kurzlebige

  - Komplexe Abhängigkeiten nicht erfassbar

Ausblick: Verteilte Spieleplattformen

- Komplexe Abhängigkeiten nicht erfassbar:
  - Zusammengehörigkeiten von Anwendungsaktionen zu Gruppen / Priorisierung einer Gruppe
  - Atomare Aktionen, “Fill or Kill”:
    Verteilte Spiele: Verlassen der Deckung nur dann möglich, wenn Handlungsfähigkeit in den 10 Sekunden danach garantiert ist
  - Kollision mit Echtzeitanforderungen
    PDA/Smartphone:
    Eingehender Anruf während der 10 Sekunden danach!
- Semantische Lücke:
  Bedeutung von Aktionen, Abstufungen