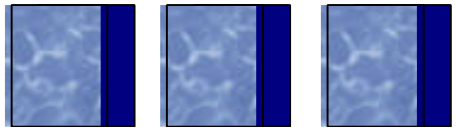
Evaluierung der QoS-Unterstützung in TAO/ACE

Großer Beleg - Zwischenstand

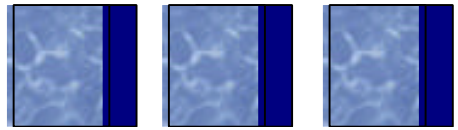
Ansgar Konermann

16. Juli 2002



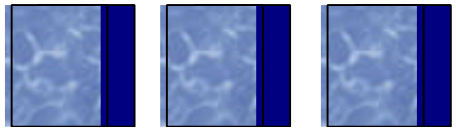
Gliederung

- Aufgabenstellung
- Echtzeitfähigkeit
- Probleme herkömmlicher ORBs
- Entwicklungsrichtlinien für RT-ORBs
- *Grober Überblick*: TAO-Architektur und RT-Fähigkeit
- Ausblick: weitere Planung



Aufgabenstellung

- Schwerpunkt: QoS-Durchsetzung in TAO/ACE ***vertikal***
- ***Was?*** Welche QoS-Eigenschaften können spezifiziert werden
- ***Wie*** erfolgt die Abbildung der QoS-Anforderungen zwischen den Schichten?
- ***Was?*** Welche Techniken zur QoS-Unterstützung gibt es?
- ***Wie*** effizient sind diese Techniken?
- ***Wo*** sind im Gesamtsystem kritische Stellen in bezug auf RT- und QoS-Fähigkeit?



"Herausforderungen"

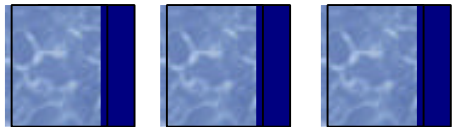
- ***Selektion*** der relevanten Forschungsberichte (ca. 380 Berichte, überwiegend "8.3" Dateinamen).
- ***Kombination*** der Informationen in diesen Berichten (teilweise widersprüchliche oder veraltete Angaben)
- ***Strukturierung*** und ***Zusammenfassung*** der Information (Detailierungsgrad?)

Echtzeitfähigkeit

- Übergeordnetes Ziel: *Planbarkeit* von Ressourcen-Nutzung (CPU, Bandbreite, ...)
- *RT-Fähigkeit notwendige Voraussetzung für QoS*
- Im TAO-Kern: QoS synonym für RT-Fähigkeit. Andere QoS-Anforderungen von CORBA-Diensten unterstützt
- Effizienz hat *häufig* positiven Einfluss auf QoS-Eigenschaften, jedoch *nicht immer!*
- Statt Festlegung auf *ein* Optimierungsziel (Effizienz ./. Vorhersagbarkeit): Nutzung von Mustern zur Flexibilisierung
- Zahlreiche *Strategien* noch sehr spät im Entwicklungszyklus wählbar (Installations- oder Laufzeit des Systems).
- *Douglas Schmidt: "[...] largely an implementation detail."*

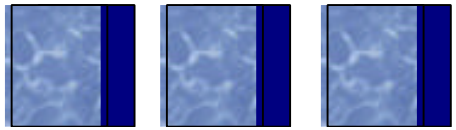
Probleme herkömmlicher ORBs

- Fehlende Mechanismen zur Spezifizierung von QoS-Eigenschaften
- Unterstützung durch das Betriebssystem und das Netzwerk nicht vorhanden bzw. wird nicht genutzt
- Kommunikationsprotokolle (z. B. hier: GIOP)
- Demultiplexierung und Verteilung von Anfragen
- Präsentationsschicht (Marshalling/Demarshalling)
- Speicherverwaltung

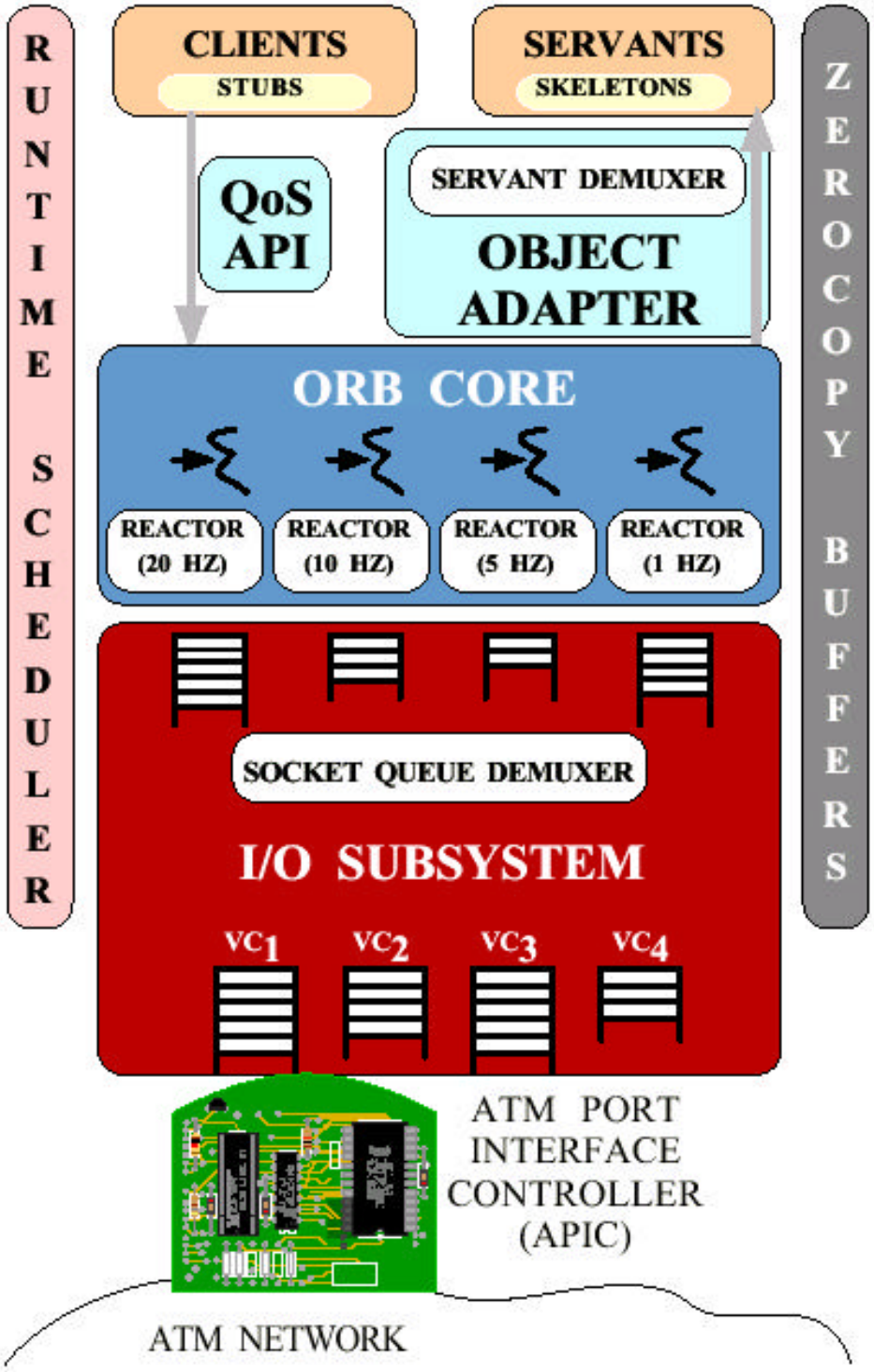


Entwicklungs-Richtlinien für RT-ORBs

- Vermeide dynamischen Verbindungsaufbau (jitter). Besser: Aufbau von Verbindungen im Voraus
- Minimiere dynamische Speicherverwaltung (locking overhead, contention, queuing): Besser: Thread Specific Storage
- Vermeide Multiplexierung von Anfragen verschiedener Priorität über eine gemeinsam benutzte Verbindung (locking overhead, priority inversion). Besser: Verbindung pro Priorität
- Parallelitätsarchitektur: flexibel, effizient und vorhersagbar. Z. B. *reactor-per-thread-priority*
- Vermeide Neuentwicklung von Betriebssystem-Mechanismen: Endpunkt-Demux, Queueing, Parallelitätssteuerung (Overhead). Besser: ORB die BS-Mechanismen nutzen lassen
- Nutze empirische Leistungsmessung zur Unterstützung des Entwurfsprozesses eines ORB-Systems

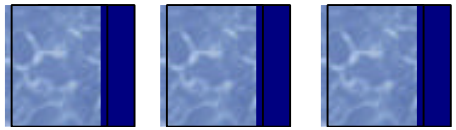


TAO-Architektur



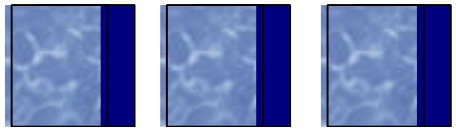
Wichtige Architekturkomponenten zur RT-Unterstützung (1)

- I/O Subsystem
  - Hochgeschwindigkeits-Netzwerk-Schnittstelle (ATM)
  - Zahlreiche RT-Optimierungen: enge Verzahnung mit ORB-Scheduler, teilweise im Kernel-Modus betrieben
  - berücksichtigt Priorität der Anfragen (vgl. Verbindungsarchitektur)
- Real-Time Scheduling Service und Runtime Scheduler
  - Mapping von "higher-level" QoS-Anforderungen auf low-level-Parameter
  - Spezifikation von: **C**: computation time (wcet), **P**: period, **D**: deadline für jede Operation.
- Speicherverwaltung:
  - Zero-Copy Buffer Management System
  - "early demultiplexing" in Shared-Memory Puffer
  - Thread Specific Storage

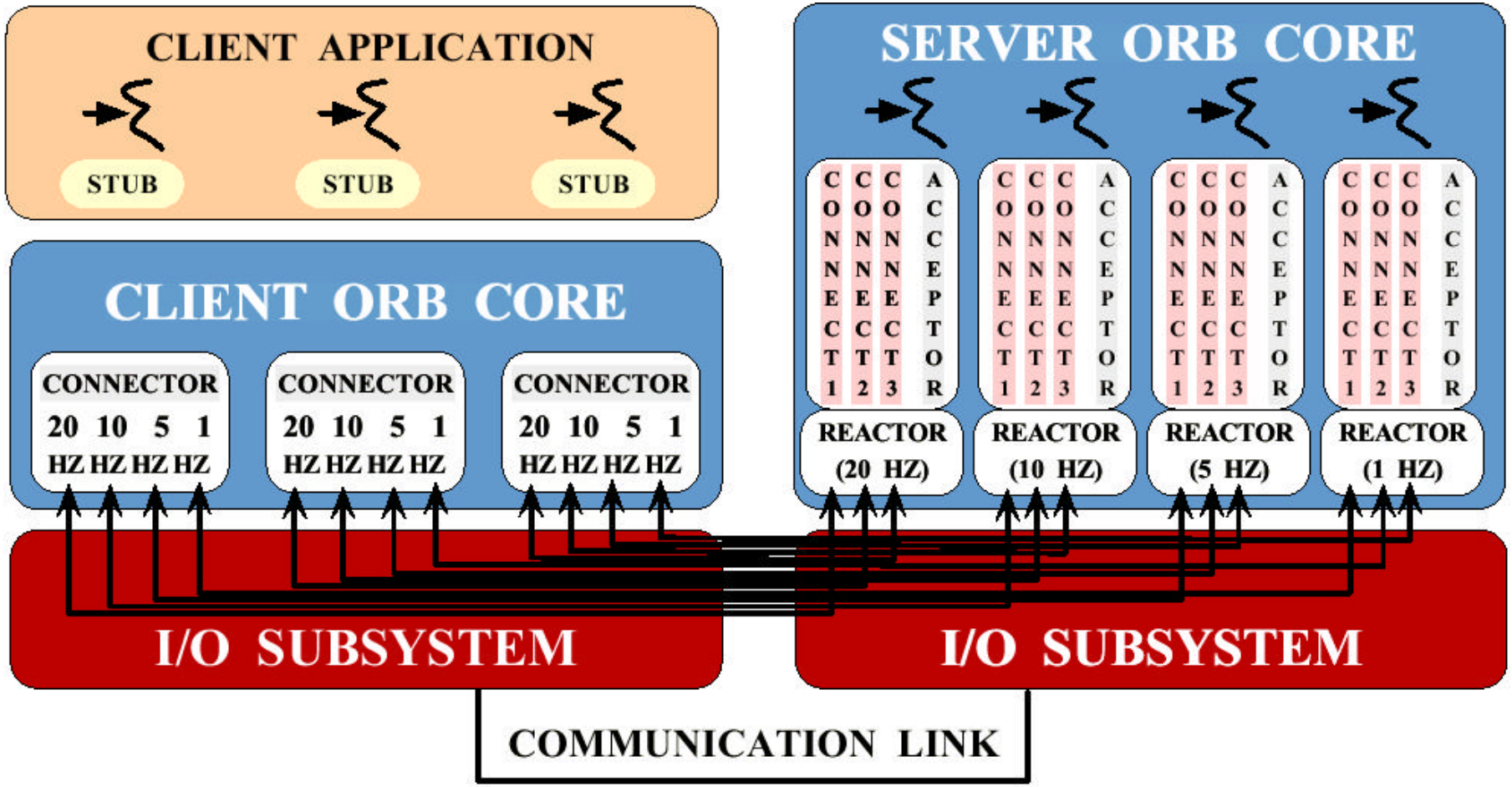


Wichtige Architekturkomponenten zur RT-Unterstützung (2)

- ORB-Kern
  - Inter-ORB-Protokollverarbeitung (GIOP)
  - Parallelitätsarchitektur ist ***prioritätsbasiert.***
  - Verbindungsarchitektur ist ***prioritätsbasiert.***
  - RIOP Echtzeit-Protokoll
  - Anpassbarkeit an Betriebssystem- und Netzwerk-Umgebungen: Nutzung von ***Mustern*** und ***Frameworks***
  - Scheduling und Dispatching: Nutzung RT-Scheduling Service

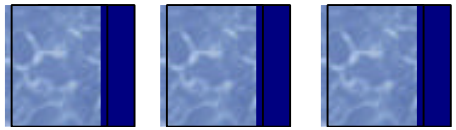


Verbindungs- und Parallelitätsarchitektur



Wichtige Architekturkomponenten zur RT-Unterstützung (3)

- Objekt-Adapter
  - Demultiplexierung in O(1): *active demultiplexing* oder *perfect hashing*
  - *Active Demultiplexing: Anfrage enthält Objekt-Schlüssel, Abbildung auf (Servant, Operation)*-Tupel über Tabelle
  - *Perfect Hashing: Nutzung einer optimalen Hashfunktion (generiert) zur Abbildung*
- *Stubs/Skeletons*
  - IDL-Compiler optimiert generierten Code
  - Vorausberechnung von Puffergrößen, Pre-Allokation von Puffern, Nutzung des Laufzeitstapels, Inlining, Caching von bereits gemarshallten Anfrageteilen



Weitere Planung

- ***Wäre ein*** *"Mechanismenkatalog zur Durchsetzung von RT-Fähigkeiten"* als Hauptergebnis der Arbeit akzeptabel?
- Übrige CORBA-Dienste: genauere Betrachtung notwendig? (derzeitiger Stand: eher nein)
- Beispielprogramm: welche Bereiche der RT-Fähigkeit von TAO sollen genauer untersucht werden?