TAO
(The **ACE ORB**)

Andrzej Boczek

Streszczenie

Wiele systemów czasu rzeczywistego używa z uniwersalnych i otwartych systemów dystrybucji obiektów, jakim jest na przykład **CORBA**. Pomimo, że **CORBA** jest odpowiednim środowiskiem dla standardowych aplikacji typu zadanie/odpowiedź, implementacje **CORBA**-y nie są jeszcze odpowiednie do zastosowań w systemach czasu rzeczywistego z powodu braku możliwości zapewnienia jakości usług (QoS) i niskiej prędkości działania.

Rysunki pochodzą z pracy autora systemu TAO, Douglasa C. Schmidta.

http://www.cs.wustl.edu/~schmidt/TAO.html

Charakterystyka CORBA-y

CORBA pozwala dokonywać operacji bez troski o:

- Fizyczną lokalizację obiektu
- Język programowania
- Platformę sprzętowej
- System operacyjny
- Protokół transportowy

A dodatkowo:

- Upraszcza rozwój oprogramowania
- Zaczyna powoli stawać się standardem komunikacji w systemach rozproszonych

Ograniczenia CORBA-y do zastosowań w systemach czasu rzeczywistego

- Brak możliwości specyfikowania wymagań jakości usług (QoS)
- Brak mechanizmów zapewniania jakości usług
- Brak wsparcia dla programów czasu rzeczywistego
- Brak optymalizacji prędkości działania

Opis projektu TAO

Celem projektu było:

- Identyfikacja rozszerzeń, niezbędnych dla zachowania QoS
- Empiryczne ustalenie cech wymaganych do zbudowania Pośrednika Żądań do Obiektów (**ORB**) czasu rzeczywistego
- Zintegrowanie strategii podsystemu wejścia/wyjscia z **ORB** dla uzyskania QoS
- Opisanie głównych "wzorców" projektowych do rozwijania, konfigurowania, rozszerzania systemów opartych na **CORBA**-ie

Douglas C. Schmidt

http://www.cs.wustl.edu/~schmidt/TAO.html

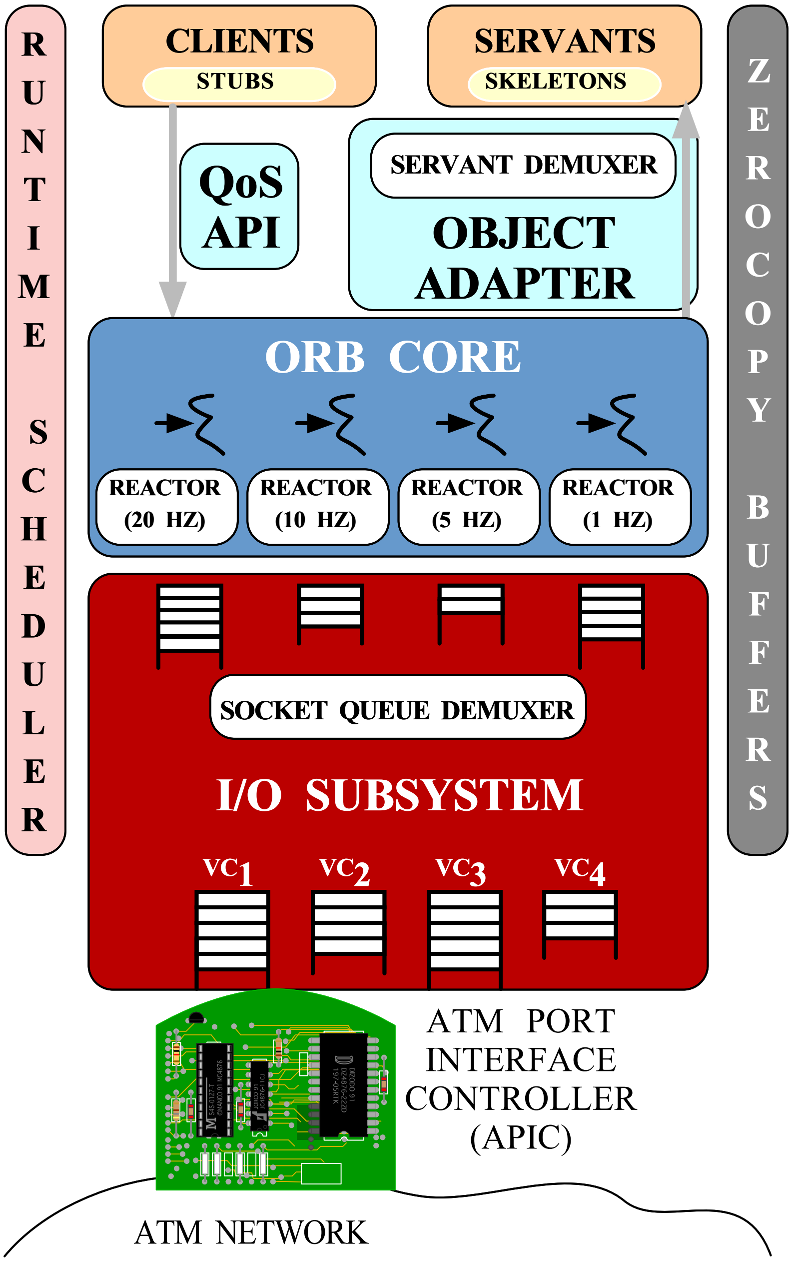
Wymagania dla systemów czasu rzeczywistego i systemów wysokowydajnych

- Polisy systemowe i mechanizmy specyfikowania wymagań jakości usług
- Wsparcie systemu operacyjnego dla zapewniania QoS
- Wydajne i przewidywalne protokoły komunikacji
- Wydajne i przewidywalne dekodowanie żądań (demultiplexing)
- Wydajna i przewidywalna warstwa prezentacji
- Wydajne i przewidywalne gospodarowanie pamięcią

Architektura systemów czasu rzeczywistego i systemów wysokowydajnych

- Podsystem wejścia/wyjścia (I/O subsystem)
- Scheduler (Run-time scheduler)
- Jądro **ORB** (**ORB** Core)
- Procesor obiektów (Object Adapter)
- Pieńki i szkielety (Stubs and skeletons)
- Zarządca pamięci (Memory manager)
- Interfejs programistyczny dla QoS

Architektura TAO



http://www.cs.wustl.edu/~schmidt/TAO.html

Podsystem wejścia/wyjścia

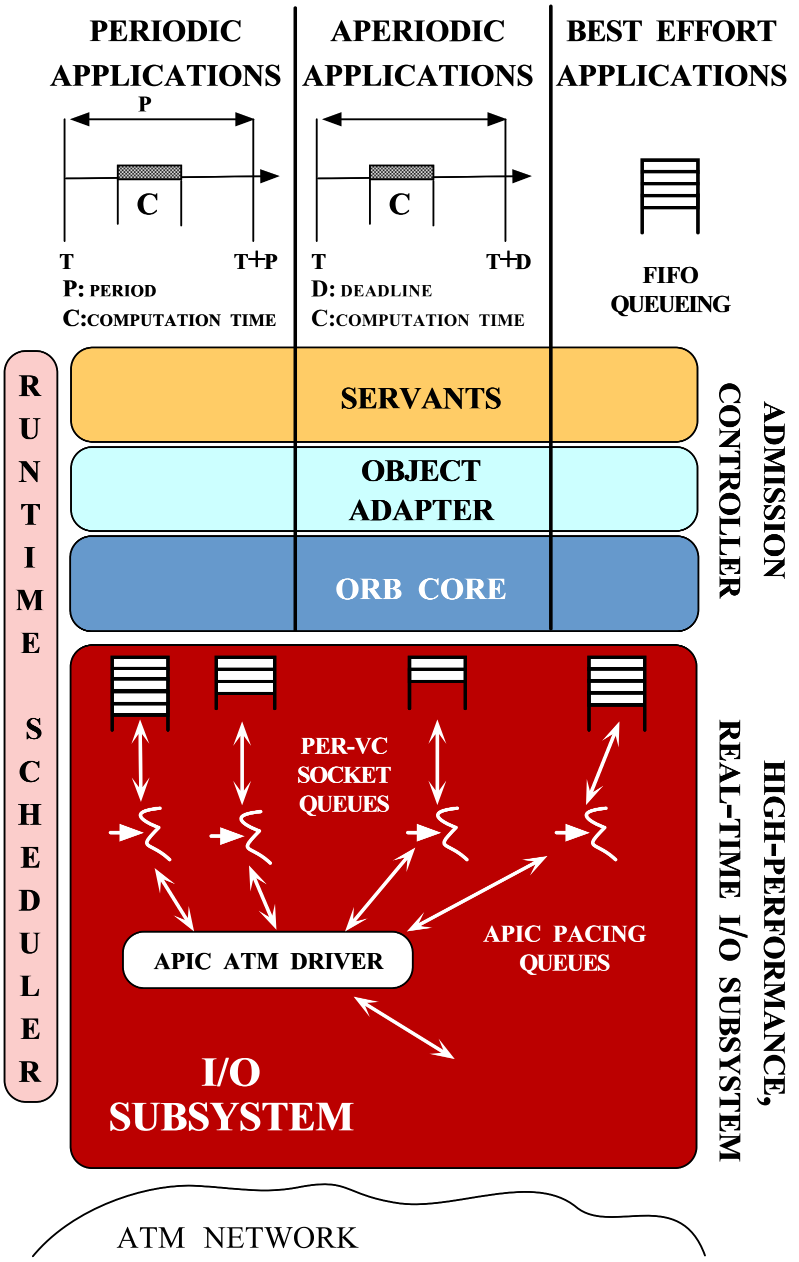
Powinien:

- Ułatwiać aplikacjom specyfikowanie wymagań jakości usług
- Minimalizować możliwość wystąpienia inwersji priorytetów i niedeterminizm
- Umożliwiać **ORB** wykożystanie cech QoS dostarczanych przez niższe warstwy sieciowe

Co jest w **TAO**:

- Szybki interfejs ATM
- Wydajny podsystem wejścia/wyjścia czasu rzeczywistego
- Scheduler czasu rzeczywistego
- Kontroler dostępu

Podsystem wejścia/wyjścia



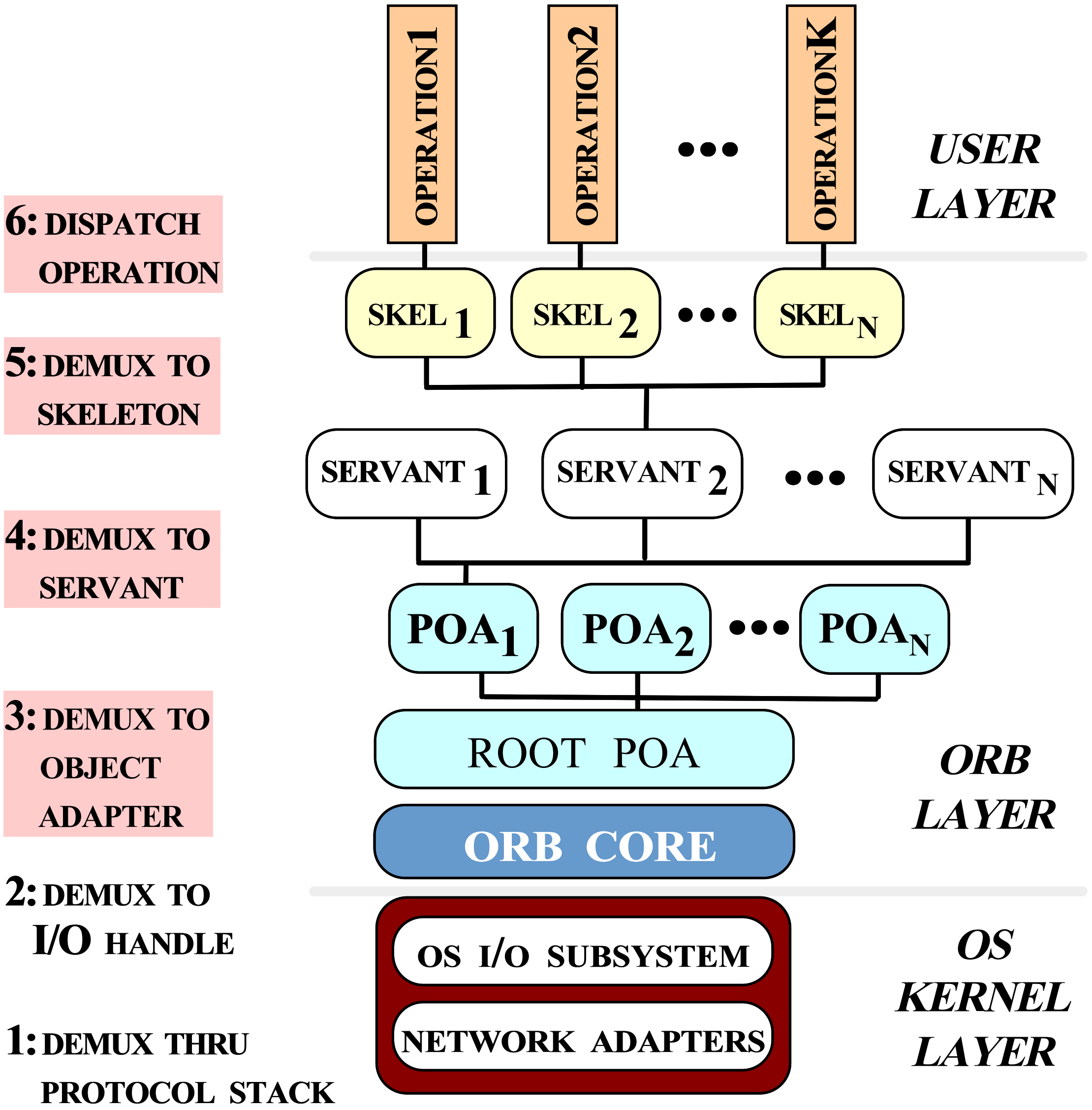
http://www.cs.wustl.edu/~schmidt/TAO.html

Jądro ORB

Dwa modele komunikacji pomiędzy **ORB**:

- Połączenia oparte na priorytecie
  - Model Connector/Reactor
  - Oddzielny kanał komunikacyjny dla każdego połączenia Connector – Reactor
  - Odzielny Reactor dla każdego priorytetu
  - Odpowiedni dla systemów planowanych statycznie
- RIOP (Real-time Inter-ORB protocol)
  - Wymagania jakości przekazywane w nagłówku żądania
  - Odpowiednie dla systemów zmiennych
  - Zgodne ze standardowym GIOP z **CORBA**-y

CORBA Object Adapters



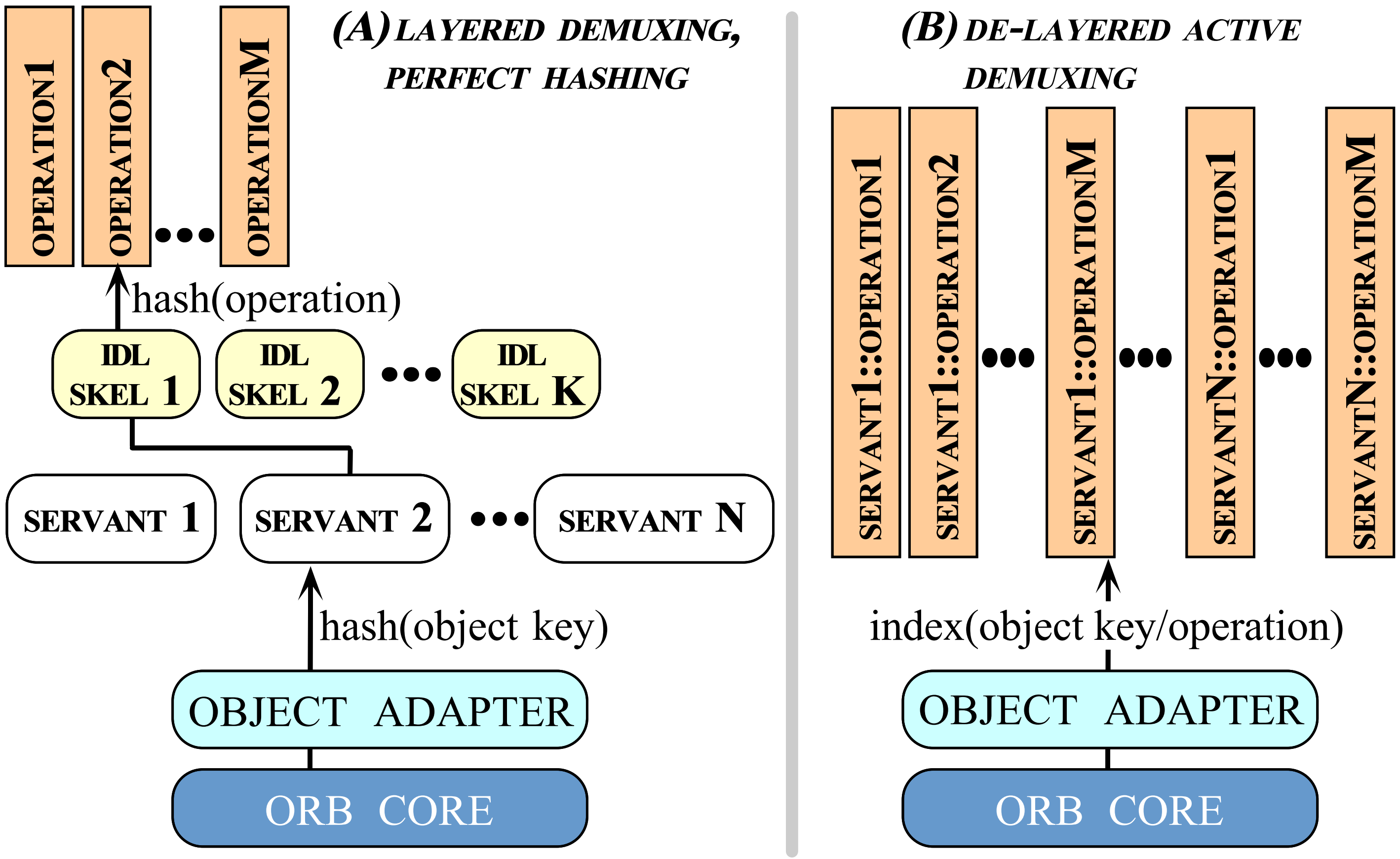
http://www.cs.wustl.edu/~schmidt/TAO.html

Procesory Obiektów (Object Adapters)

Dwa rodzaje dekodowania (demultiplex) obiektów.

- Za pomocą funkcji perfekcyjnie hashującej
  - Identyfikatory obiektów znane są a priori
  - Generowana jest funkcja perfekcyjnie hashująca (GPERF)
  - Żądanie jest dekodowane w czasie stałym $O(1)$
- Aktywne dekodowanie
  - Każdy servant w momencie rejestracji ma przydzielany indeks w tablicy
  - Klient poszukując servanta dostaje również jego indeks
  - Żądanie jest dekodowane w czasie stałym $O(1)$

TAO Object Adapters



http://www.cs.wustl.edu/∼schmidt/TAO.html

Pieńki i szkielety

Optymalizacja na poziomie kompilacji IDL:

- Redukcja używania pamięci dynamicznej – alokacja a priori wystarczającej ilości pamięci na dane
- Redukcja kopiowania danych – kopiowanie blokami
- Redukcja kosztu wołania małych pieńków (inline optimization)

Zarządzanie pamięcią

Optymalizacja zarządzania pamięcią w **TAO** idzie w kierunku zmniejszenia zapotrzebowania **ORB** na pamięć:

- Bufory “Zero-copy”
- Bezpośrednie współdziałanie z szyną systemową i urządzeniami I/O
- Przyjmowanie danych od urządzenia wprost do pamięci dzielonej między poziomem jądra a poziomem użytkownika
- Możliwość pobierania pamięci dla wątku z dedykowanej sterty (heap)

Interfejs programistyczny

```
struct RT_Info
{
  Time worst_case_execution_time;
  Time typical_execution_time;
  Time cached_execution_time;
  Period_t period;
  Criticality_t criticality;
  Importance_t importance;
  Quantum_t quantum;
  Dependency_Set dependencies;
  OS_Priority priority;
};
```