

QoS enabled OO Middleware

**Vishal Kachroo, Yamuna Krishnamurthy
and Fred Kuhns**

{vishal,yamuna,fredk}@cs.wustl.edu.edu

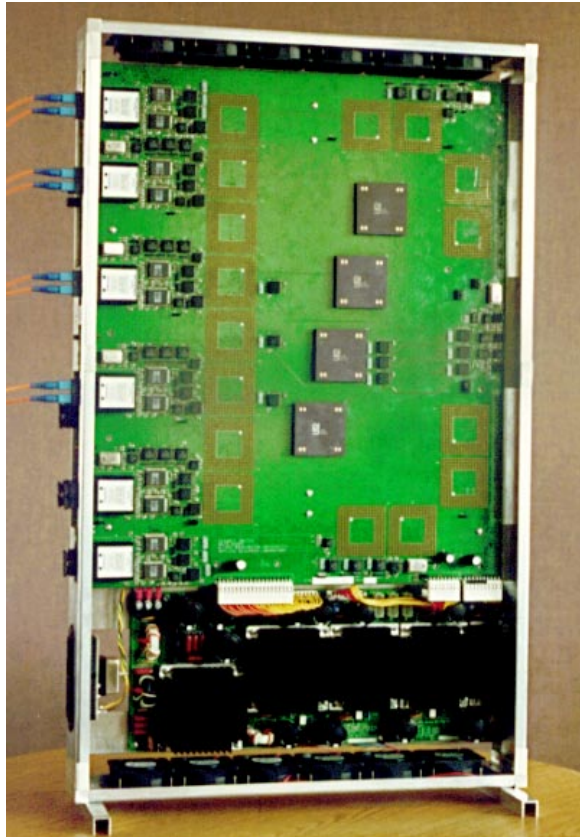
Center for Distributed Object Computing
Department of Computer Science
Washington University
St Louis Mo.



Sponsors

Motorola, Boeing

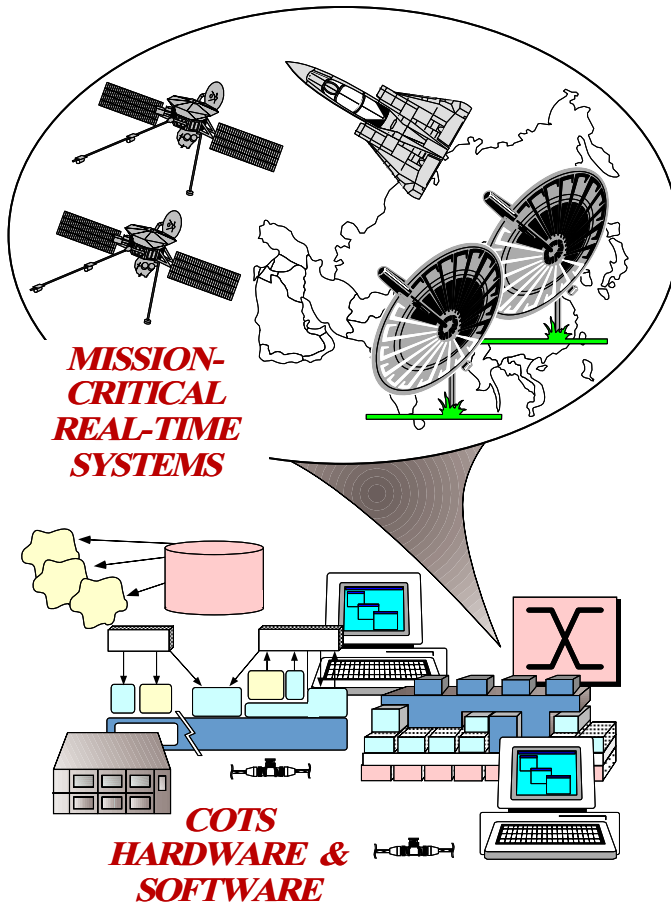
Motivation: the QoS-enabled Software Crisis



www.arl.wustl.edu/arl/

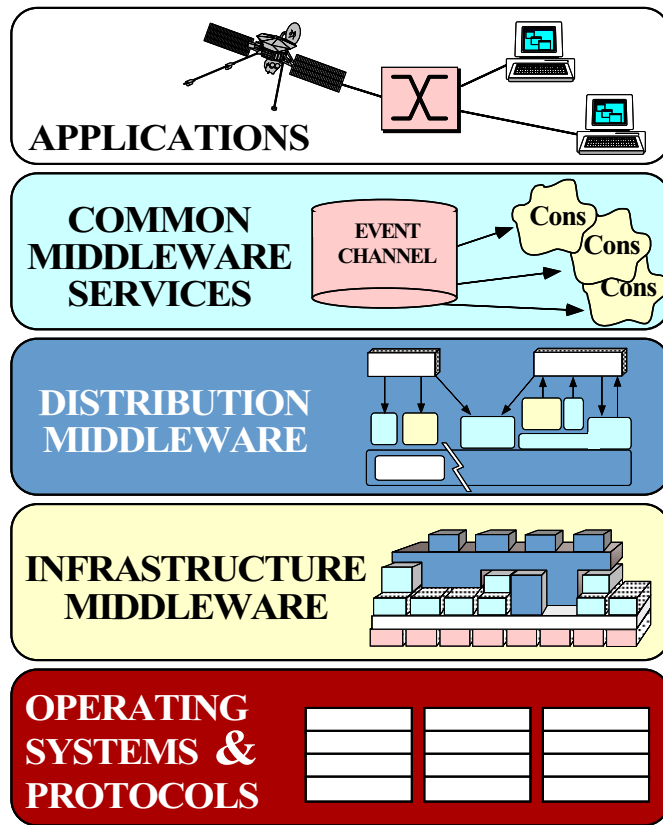
- **Symptoms**
 - Communication *hardware* gets smaller, faster, cheaper
 - Communication *software* gets larger, slower, more expensive
- **Culprits**
 - *Inherent* and *accidental* complexity
- **Solution Approach**
 - *Standards-based COTS Hardware & Software*

Problem: the COTS Hardware & Software Crisis



- Context
 - Adopting **COTS hardware & software** is increasingly essential for real-time mission-critical systems
- Problems
 - **Inherent** and **accidental** complexity
 - **Integration** woes
- Solution Approach
 - **Standards-based adaptive COTS middleware**

Context: Levels of Abstraction in Software



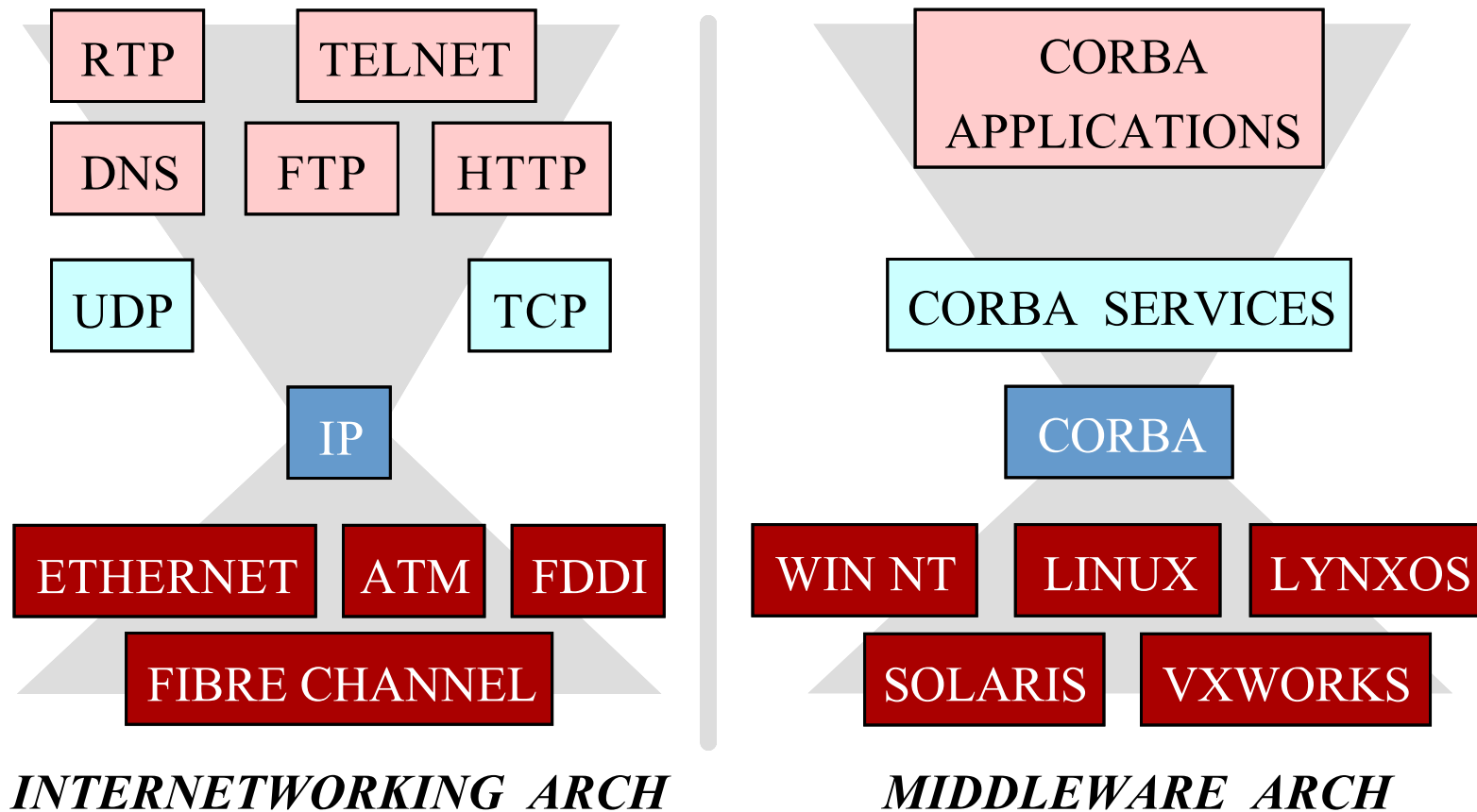
• Observations

- Historically, distributed apps built directly atop OS
- Today, more and more apps built atop *middleware*
- Middleware has several layers

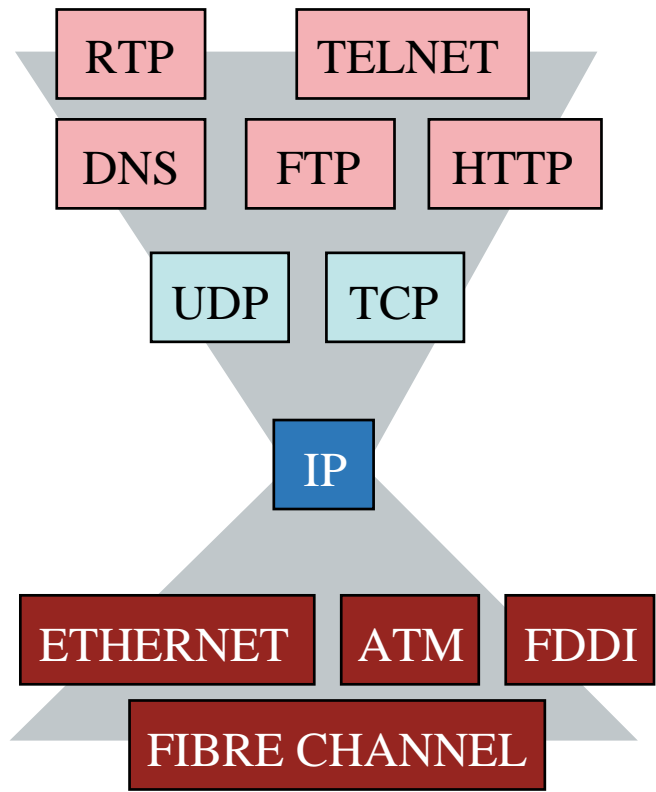
• Challenges

- Buy vs. build
- Identify reuse boundaries
- Determine where to add value

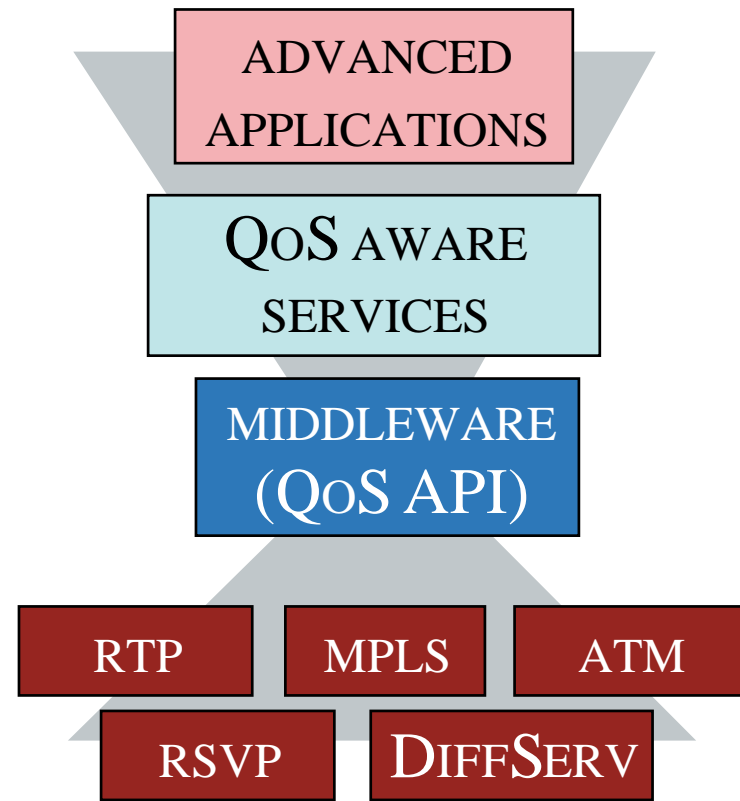
Context: Levels of Abstraction in Internetworking and Middleware



Solution: QoS Enabled Middleware

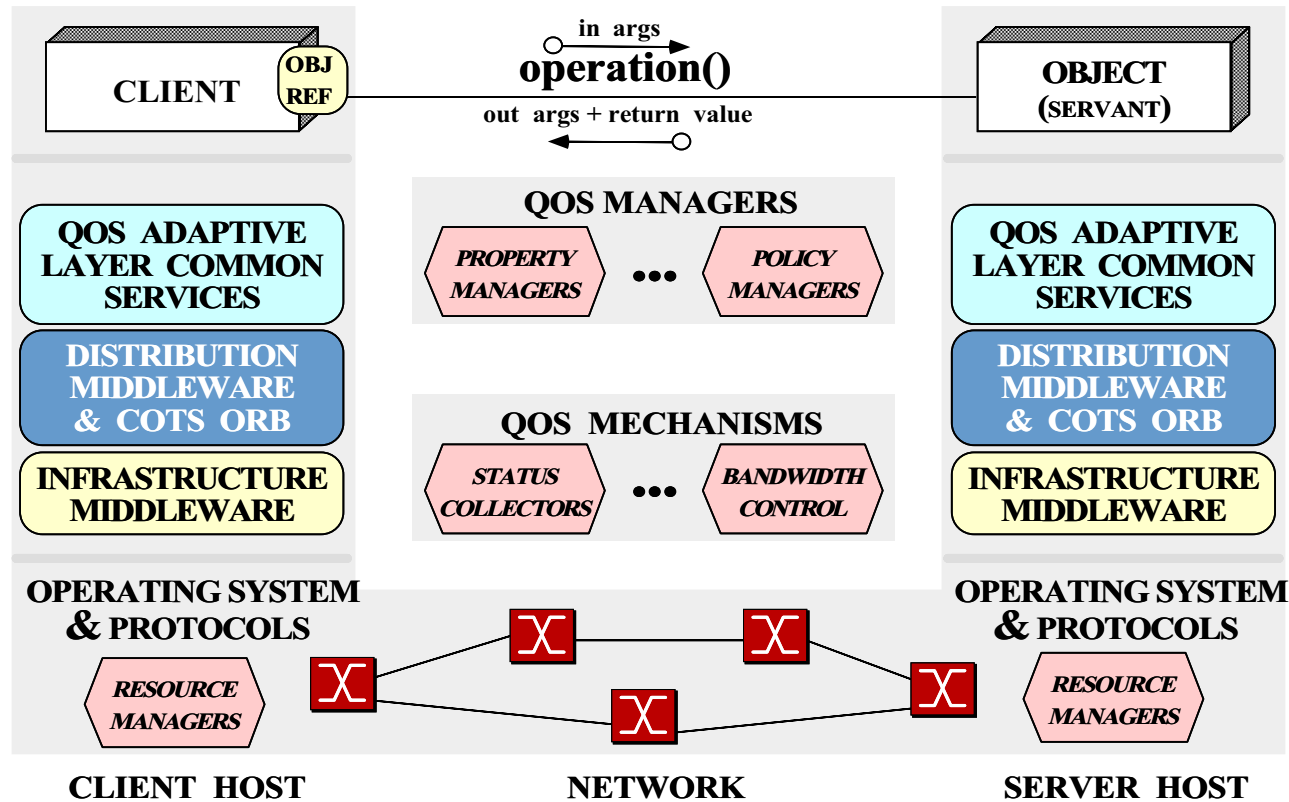


INTERNETWORKING ARCH

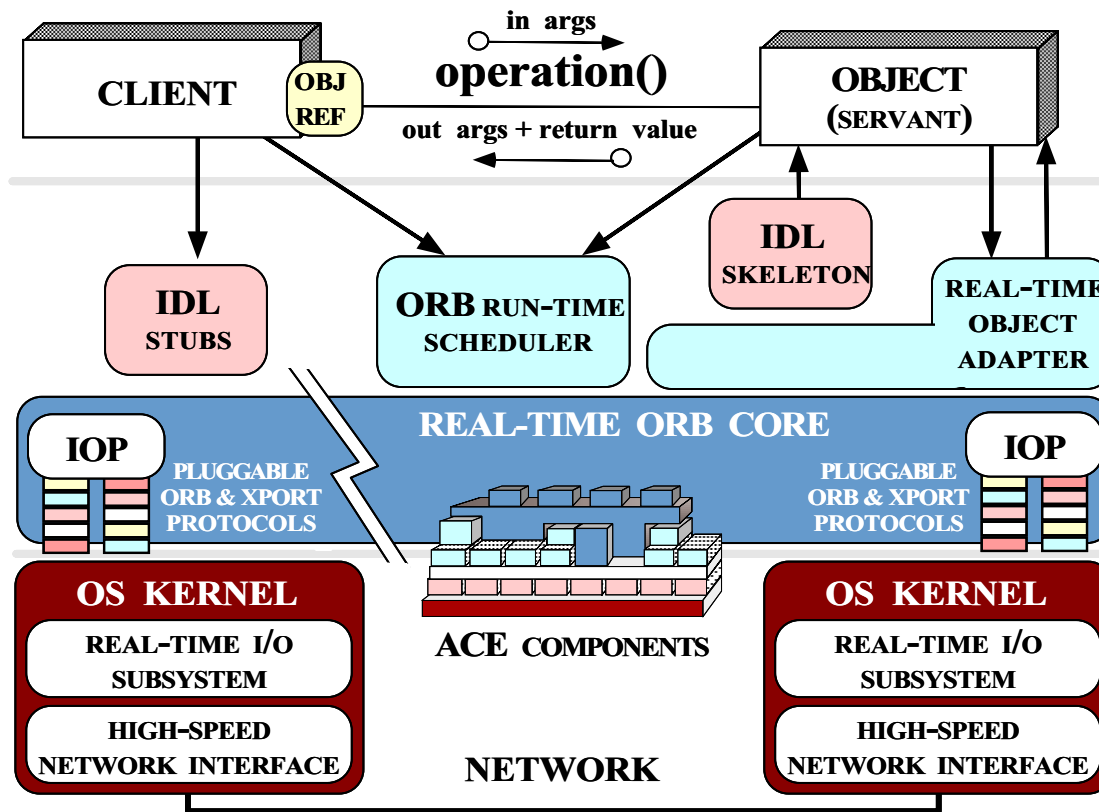


QoS ENABLED MIDDLEWARE ARCH

Creating a Framework to Support QoS Enabled Middleware



Our Approach: The ACE ORB (TAO)

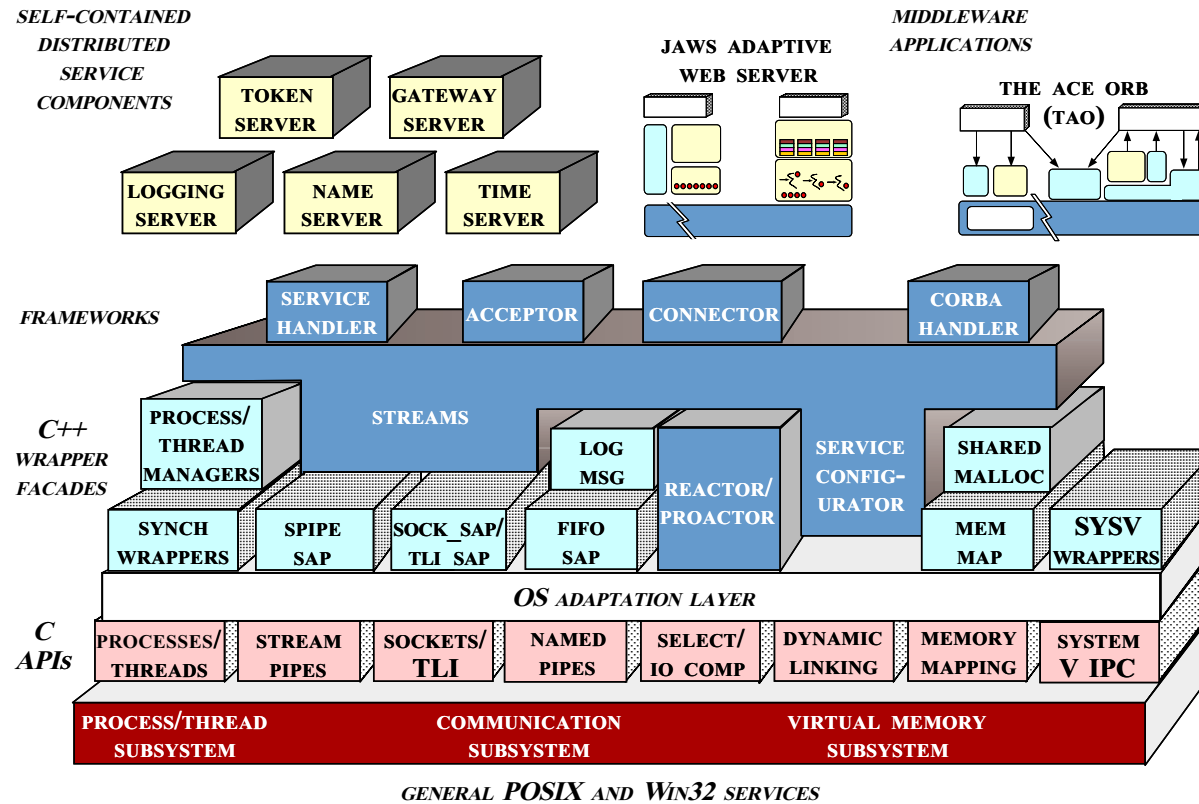


www.cs.wustl.edu/~schmidt/TAO.html

TAO Overview →

- An open-source, standards-based, real-time, high-performance CORBA ORB
- Runs on POSIX/UNIX, Win32, & RTOS platforms
 - e.g., VxWorks, Chorus, LynxOS
- Leverages ACE

The ADAPTIVE Communication Environment (ACE)



ACE Overview →

- A concurrent OO networking framework
- Available in C++ and Java
- Ported to POSIX, Win32, and RTOSs

Related work →

- x-Kernel
- SysV STREAMS

www.cs.wustl.edu/~schmidt/ACE.html



ACE and TAO Statistics

- Over 50 person-years of effort
 - ACE > 200,000 LOC
 - TAO > 200,000 LOC
 - TAO IDL compiler > 130,000 LOC
 - TAO CORBA Object Services > 150,000 LOC
- Ported to UNIX, Win32, MVS, and RTOS platforms
- Large user community
 - ~schmidt/ACE-users.html
- Currently used by dozens of companies
 - Bellcore, BBN, Boeing, Ericsson, Hughes, Kodak, Lockheed, Lucent, Motorola, Nokia, Nortel, Raytheon, SAIC, Siemens, etc.
- Supported commercially
 - ACE → www.riverace.com
 - TAO → www.theaceorb.com

Goals of the ACE QoS Project

- Short term
 - Develop a small application using GQoS APIs
 - Develop a small application using RAPI APIs
 - Identify common patterns between the two implementations
 - Design ACE APIs for QoS that abstract out RAPI and GQoS
 - Implement the ACE QoS APIs based on the above design
 - Write a test for the ACE QoS APIs
- Long term
 - Expose the ACE QoS APIs through TAO
 - Use Pluggable Protocols to indicate QoS to the lower layers
 - Use the ACE QoS APIs in the TAO Audio/Video Service

Targeting RSVP

- Internet control protocol like IGMP
- Signalling not a Routing Protocol
- Based on the IntServ Model
- Unicast as well as many-to-many multicast
- Receiver initiated
- Simplex
- Soft state in routers
- Two implementations
 - RAPI (RSVP API, an Implementation on UNIX)
 - GQoS (Generic QoS, a Win2K implementation)

GQoS API / Data Structures Overview

- WSASocket ()
- WSAGetEnumProtocols ()
- WSAIoctl ()
- WSAJoinLeaf ()
- WSAAccept ()
- WSAConnect ()

- QoS Structures:

```
// QOS Structure.
struct QOS {
    FLOWSPEC SendingFlowspec;
    FLOWSPEC ReceivingFlowspec;
    WSABUF ProviderSpecific
}

// struct FLOWSPEC
struct FLOWSPEC {
    int32 TokenRate;
    int32 TokenBucketRate;
    int32 PeakBandwidth;
    int32 Latency;
    int32 DelayVariation;
    SERVICETYPE ServiceType;
    int32 MaxSDUSize;
    int32 MinimumPolicedSize;
}
```

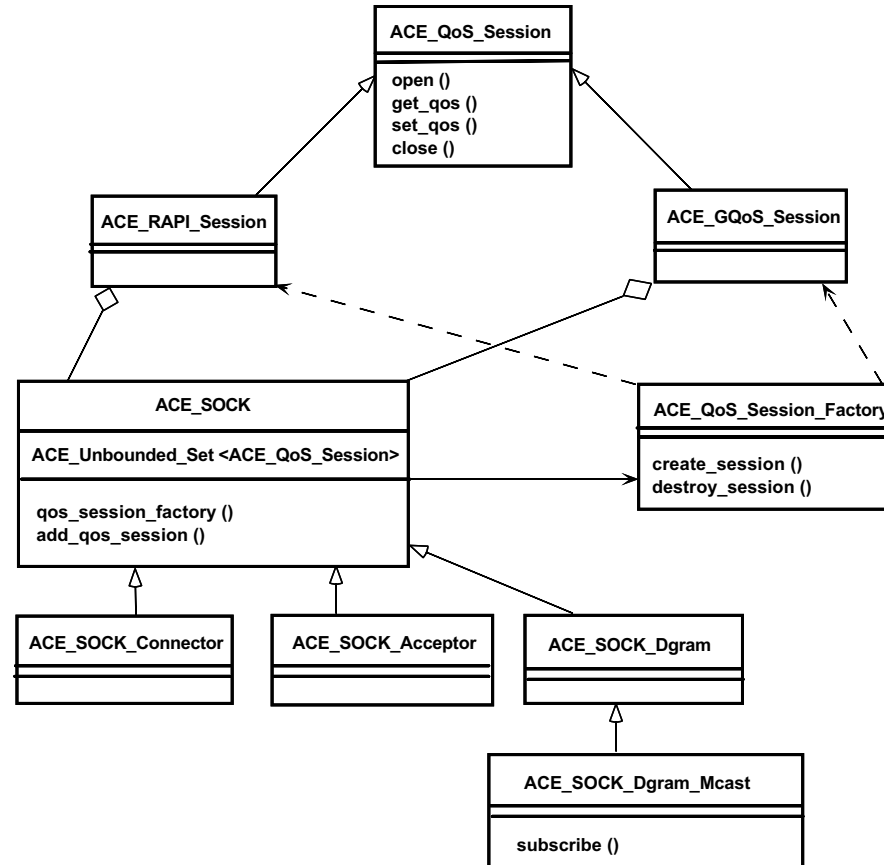
RAPI API / Data Structures Overview

- rapi_reserve () : Session {host,port,protocol_id}
- rapi_sender () : Sender reservations
- rapi_reserve () : Receiver reservations
- rapi_release () : End session
- RAPI Objects : TSpec, FilterSpec, AdSpec
- Events : PATH, RESV, PATH_ERROR, RESV_ERROR, RESV_CONFIRM

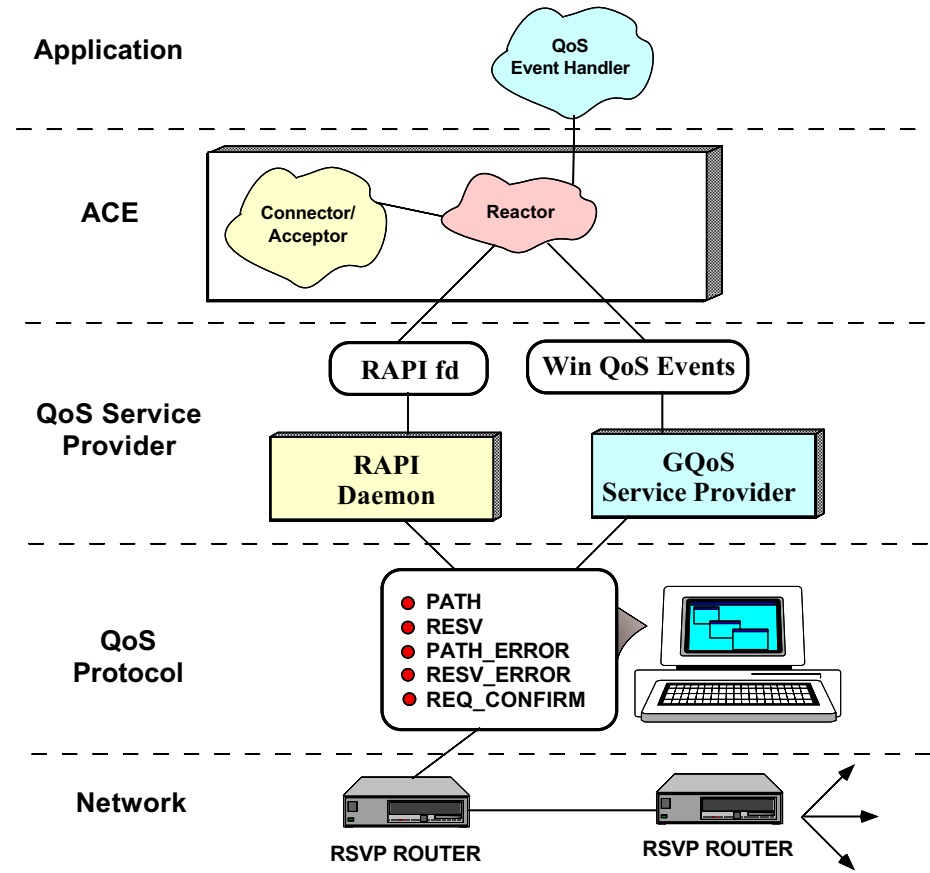
ACE QoS API Overview

- Unified view of different QoS technologies
- Portability, QoS Parameters, Extensibility.
- Wrappers for low level QoS APIs
- Notion of a QoS session
- Handling QoS events through the ACE Reactor
- Limitations because of generalization

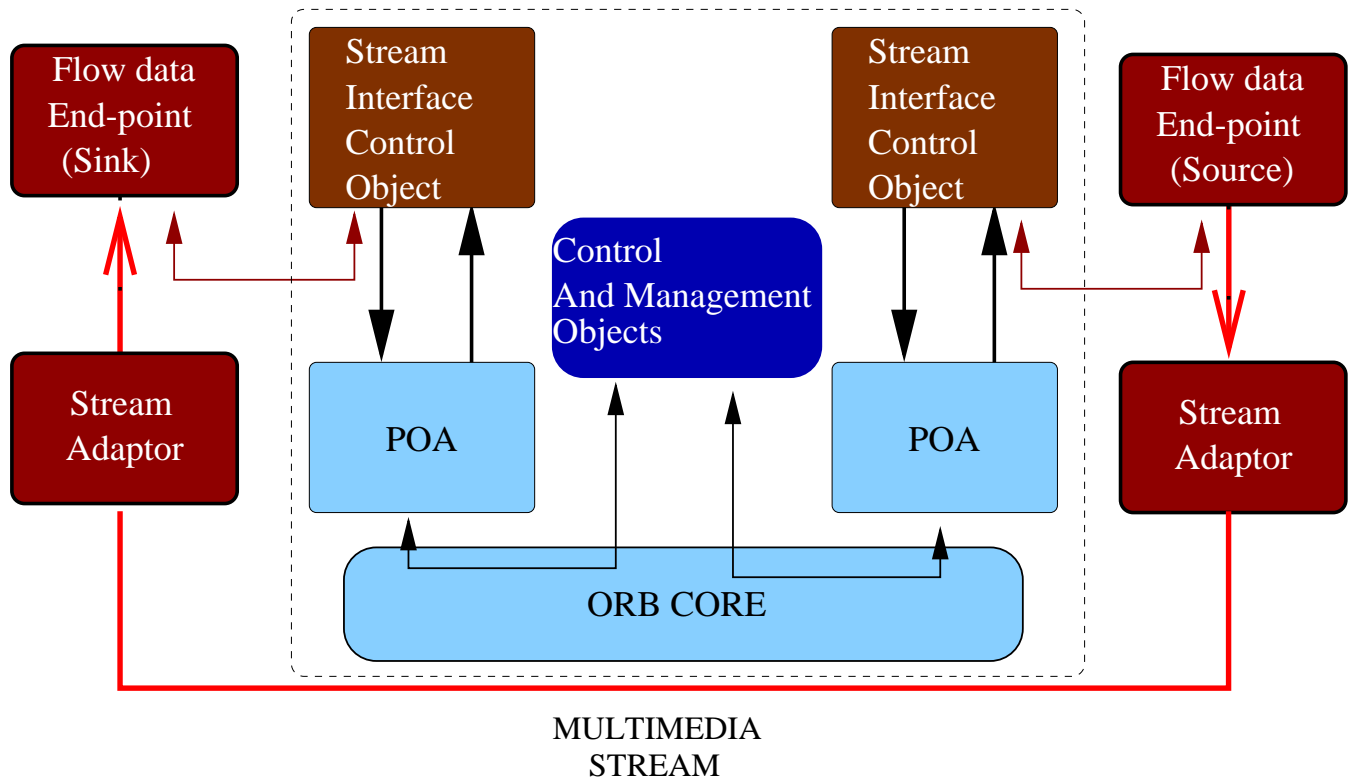
QoS API Class Diagram



Event Notification



CORBA Audio/Video Streaming



www.cs.wustl.edu/~schmidt/av.ps.gz

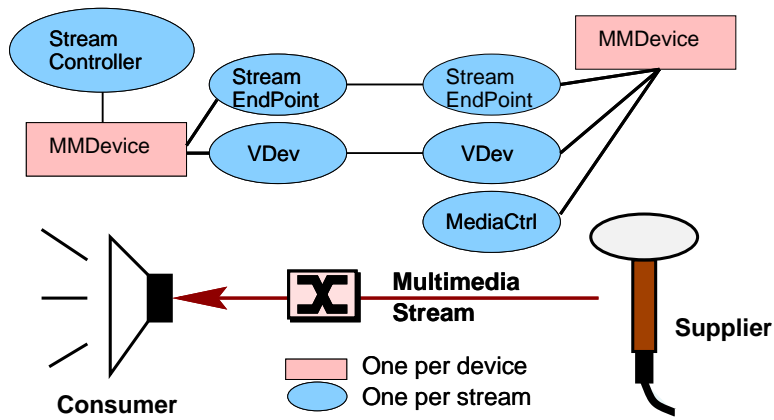
Flexibility

- Uses CORBA for control messages and properties

Efficiency

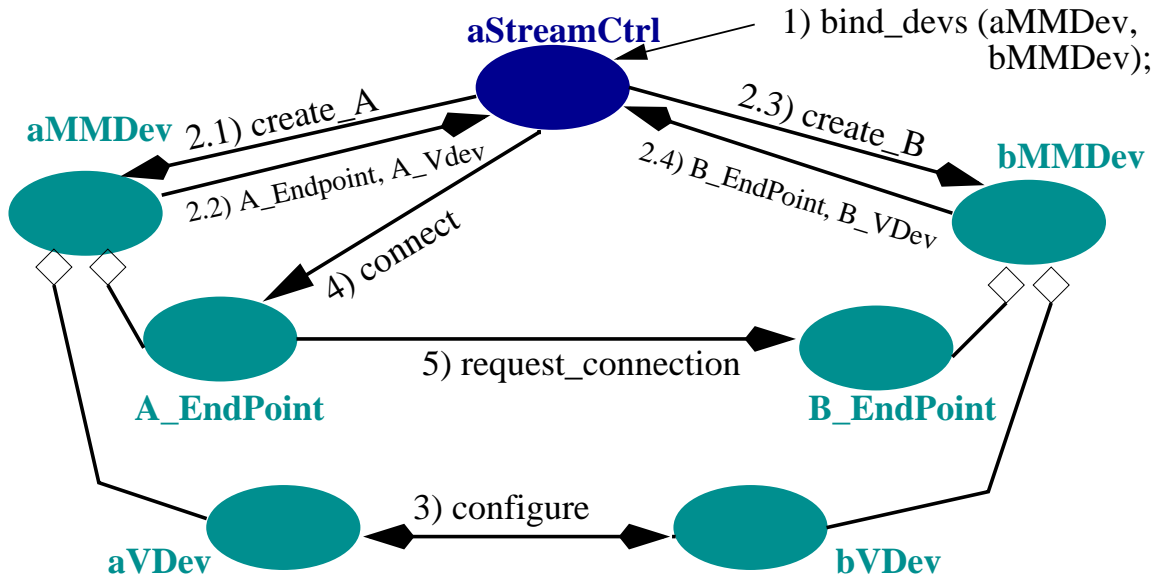
- Out of band transport connection for high throughput

A/V Service Overview - Stream interfaces



- **StreamCtrl** - Controlling the Stream i.e start/stop.
- **MMDDevice** - interface for a logical or Physical device. eg. VideoPhone, Video-on-Demand Server.
- **StreamEndPoint** - Network specific aspects of a stream.
- **StreamEndPoint_A/B** - Provides MultiPoint source/sink operations.
- **VDev** - Properties of the Stream eg. No.of flows.
- **MCastConfigIf** - Container for Multipoint sink VDevs.

Stream Establishment



- a=consumer, b=supplier
- Consumer creates its own MMDevice and gets supplier MMDevice
- Consumer uses the StreamCtrl to *bind* the two MMDevices

Need for QoS in A/V Service

- Predictable delivery at constant bit rate.
- Delay tolerance
 - Asynchronous
 - Synchronous
 - Interactive
- Difference in operational requirements.
- Adaptive resource allocation and management.

A/V Service Spec Support for QoS

- Two levels of specifying QoS - Application (eg. audio sample, video framerate) and Network (eg. service type, bandwidth) levels.
- QoS definition is a named list of properties and their values.
- `modify_qos` (`newQoS`) method for QoS renegotiations.

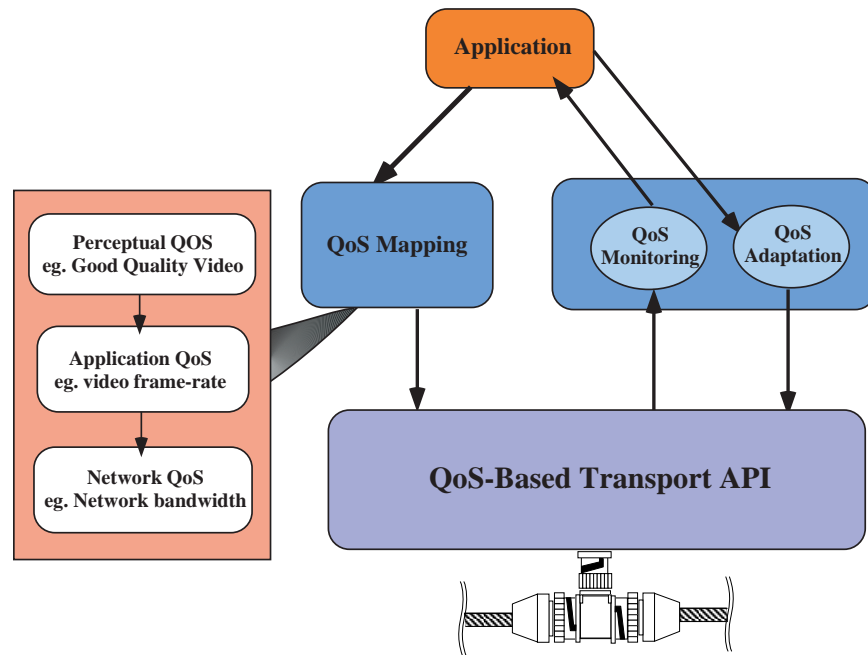
- IDL for QoS Structures:

```
// From Property Service
typedef string PropertyName;

struct Property{
    PropertyName property_name;
    any property_value;
};
typedef sequence<Property> Properties;

struct QoS{
    string QoSType;
    Properties QoSParams;
};
typedef sequence<QoS> streamQoS;
```

A/V Service QoS Framework Components



- QoS Mapping
- QoS Monitoring
- QoS-Based Transport API

QoS Mapping

- Translation of QoS specifications between different levels eg. between application and network levels.
- Reserve network resources at connection establishment.
- Good mapping rules to avoid reservation of too much (or too little) resources.
- QoS service and parameter mapping.
- Required both at connection establishment and renegotiation time.

QoS Monitoring and Adaptation

- **QoS Monitoring**

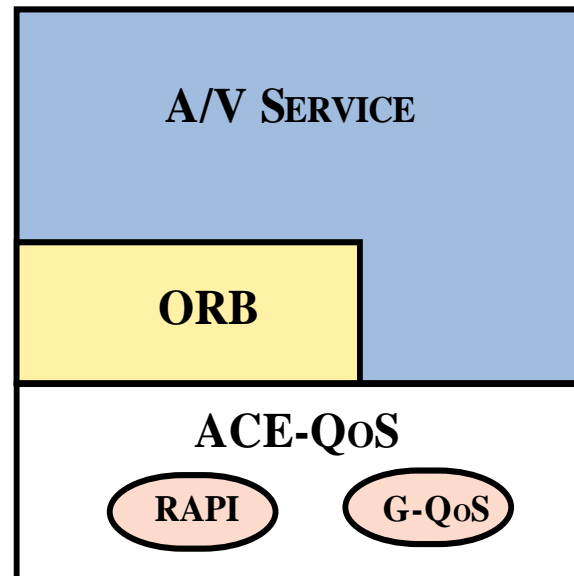
- Mechanism for measuring end-to-end QoS parameters over a finite time period.
- Typically done on the receiving side.
- Notification of QoS changes and violations to the application through feedback channels.

- **QoS Adaptation**

- Take actions based on the measured QoS and the application QoS requirements.
- Typically done on the sending side.
- Adaptation can be at the transport (eg. flow control), application (eg. MPEG-II coding rate adaptation) and at the signalling (eg. QoS renegotiation) levels.

QoS-Based Transport API

- Provides calls for provisioning, control (renegotiation and violation notification) and media transfer.
- ACE-QoS API's provide the required QoS-based transport API.



Design Challenges

- Defining generic QoS mappings for various flows.
- Designing a flexible and extensible QoS monitoring and adaptation framework.
- Understanding QoS specifications for different flow protocols.

Concluding Remarks

- Motivation to develop a unified QoS API
- Common abstractions between different QoS mechanisms
- Leveraging the existing QoS implementations
- Pluggability of new QoS implementations
- Using the existing ACE infrastructure
- Using Design Patterns
- Finally, an implementation that works !!