AQUILA

AQUILA (IST-1999-10077)

Adaptive Resource Control for QoS Using an IP-based Layered Architecture

Project Defense

"Development of C++ Client for a Java QoS API based on CORBA"

Sandeep Misra

TECHNISCHE UNIVERSITÄT DRESDEN

http://www-st.inf.tu-dresden.de/aquila/

Outline

- AQUILA Architecture
- Objective
- CORBA : Common Object Request Broker Arch.
- ORB’s : Object Request Broker
- ORBacus Guide
- The C++ Client
- Tests
- Summary

AQUILA

Best Effort Traffic

- Current traffic flow : Best effort
- Requirements :

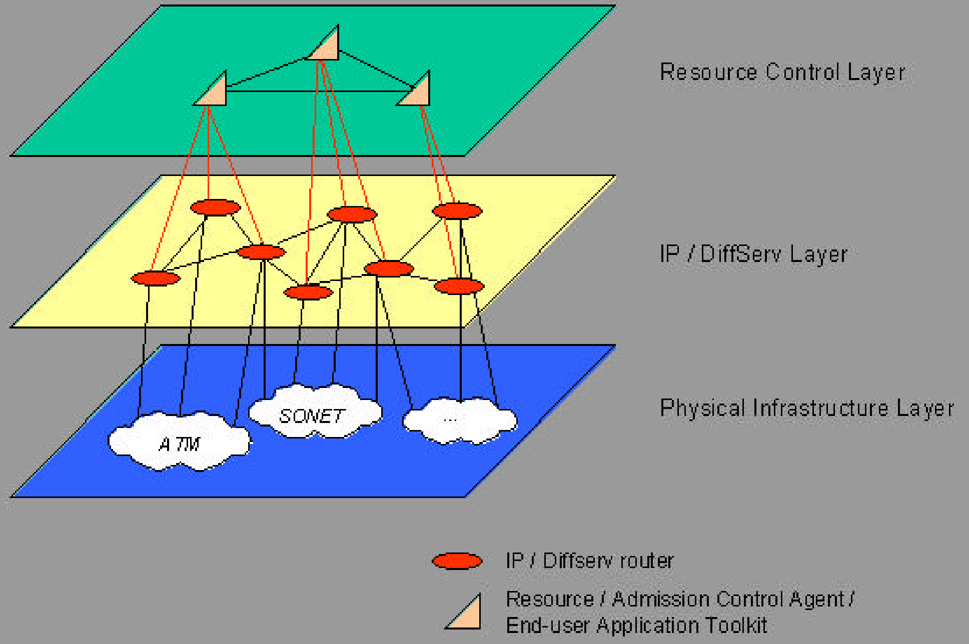
  1. Traffic Classification

  2. Real Time Traffic

  3. IP Convergence
- User is paying for the service
- Need for QoS

AQUILA Architecture

- Differentiated Services (Diffserv)
- AQUILA network architecture is based on the DiffServ network concept


Resource Control Layer
IP / DiffServ Layer
Physical Infrastructure Layer
ATM
SONET
...
IP / Diffserv router
Resource / Admission Control Agent / End-user Application Toolkit

AQUILA Components

- RCA : Resource Control Agent

  Monitors and controls the available resources in the network.

  Based on the prior history of resource usage and actual requests, the RCAs distribute resource shares to the Admission Control Agents.

- ACA : Admission Control Agent

  Performs policy and admission control at the network edges

  Each ACA gets a certain amount of resources by the RCA enabling the ACA to perform admission control autonomously

AQUILA Component : The EAT

- **EAT : End-user Application Toolkit**

  The EAT is a middleware architecture for the transparent support of QoS for applications and their users

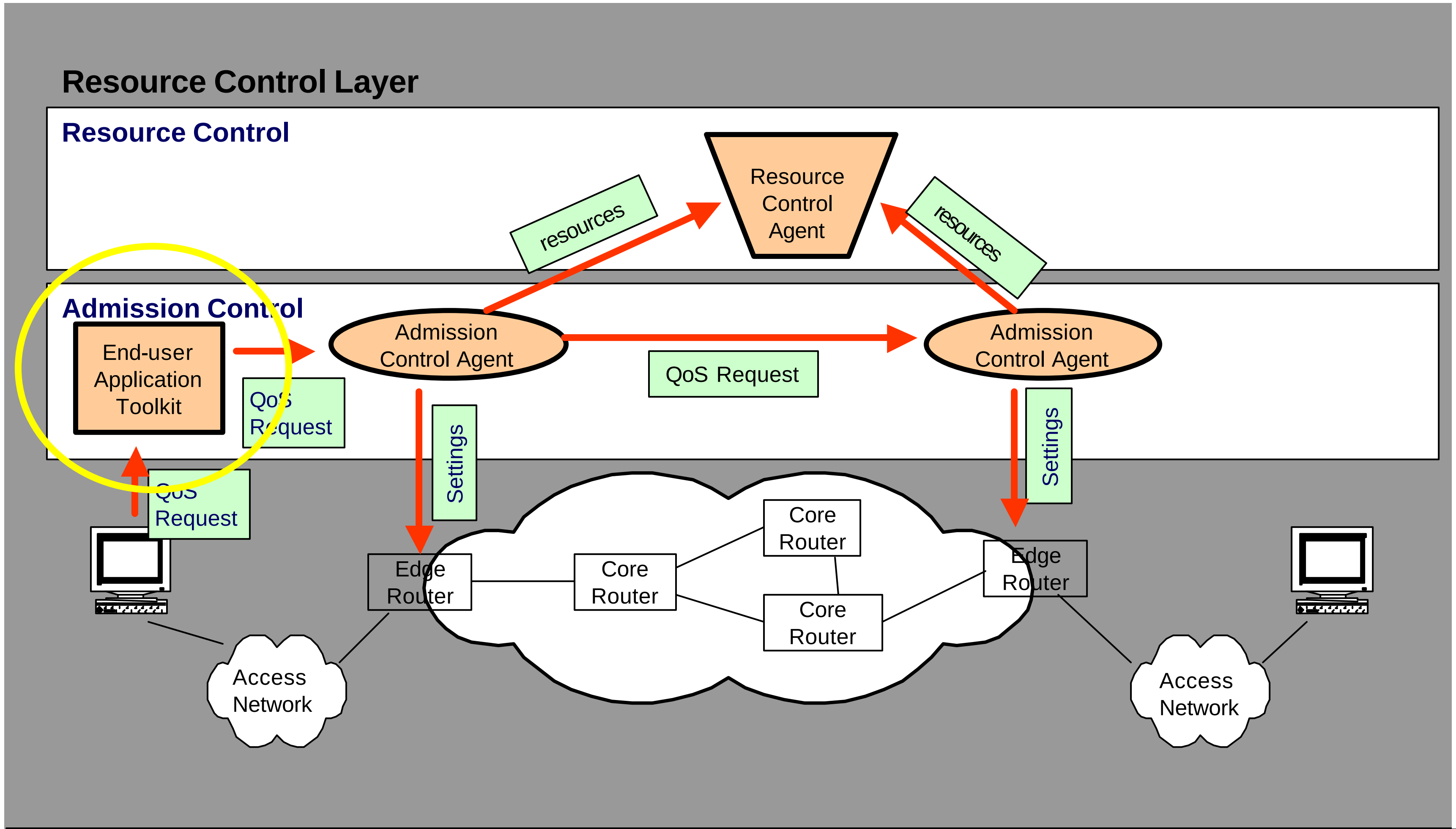
  The EAT provides an efficient mechanism for the presentation of available QoS options to the user, for the selection of the appropriate quality parameters and the transmission of a reservation request to the RCL.

- **EAT Application Programming Interface**

  The Application Programming Interface (API) of the EAT provides application services /interfaces for QoS

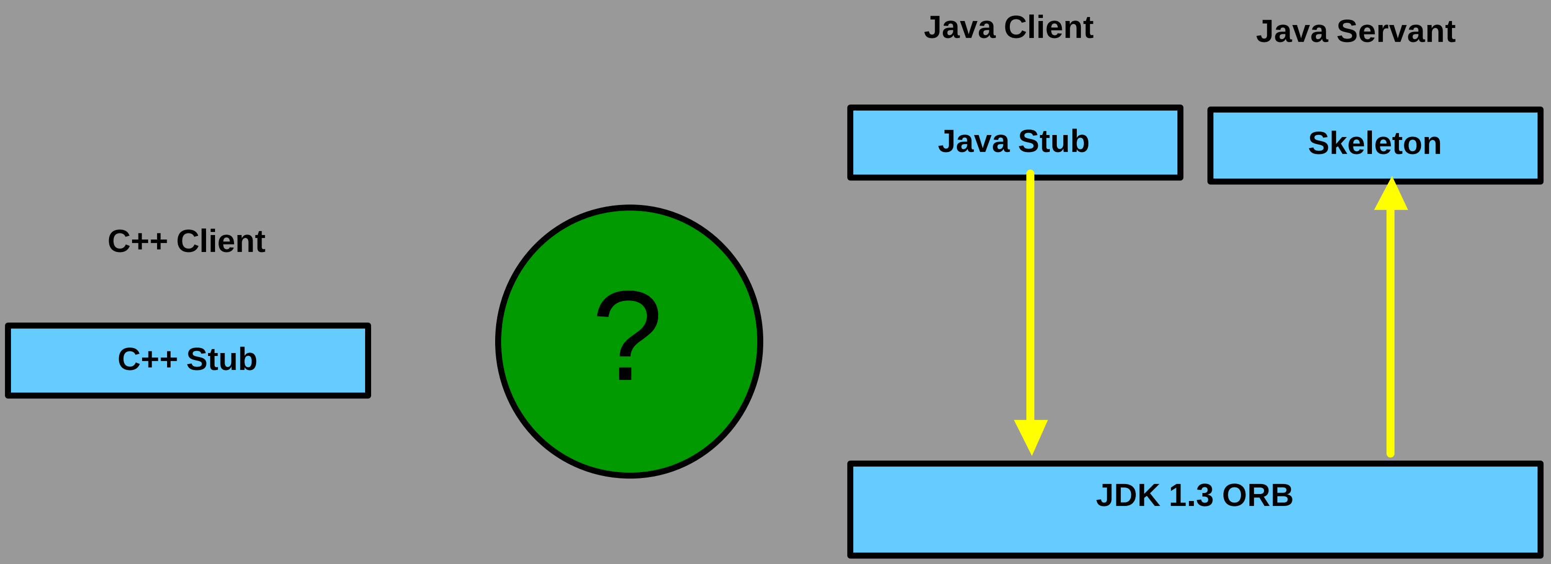
  With help of this API, application and service developers can directly access the QoS features of AQUILA's

AQUILA Architecture



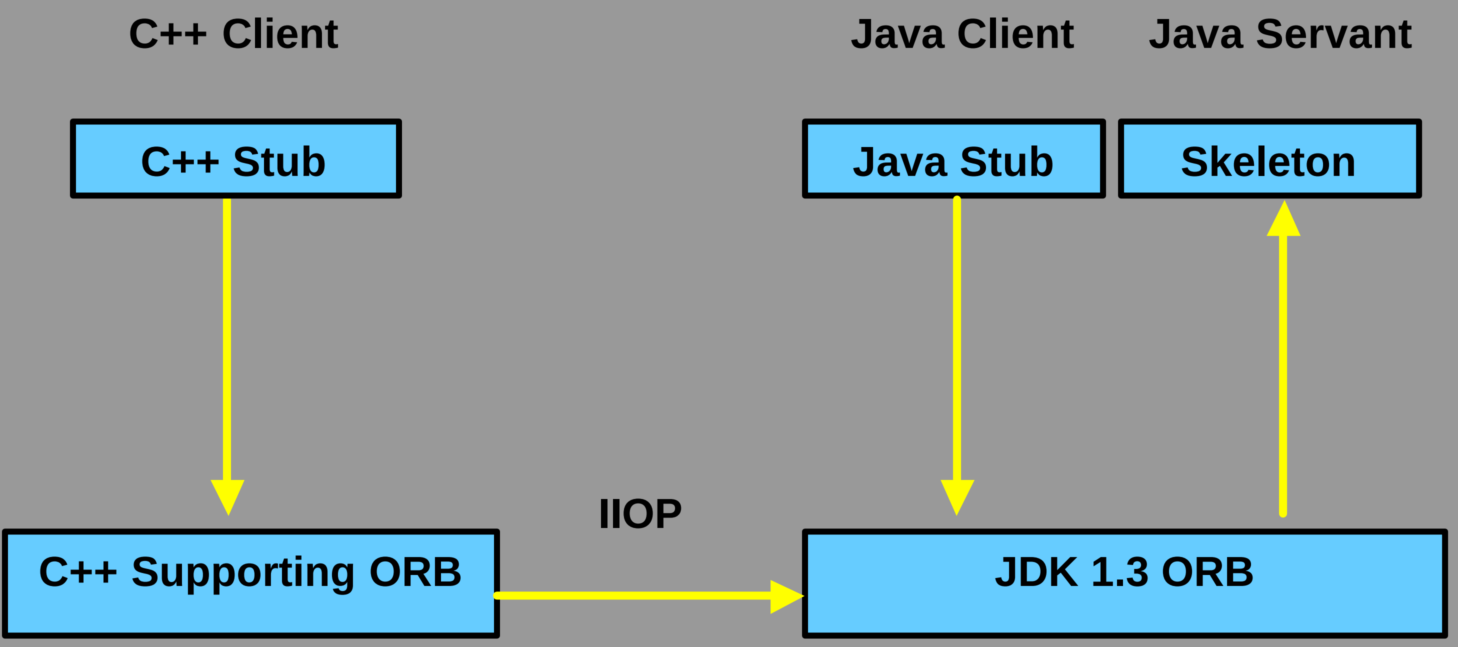
- **AQUILA**
- Objective
- CORBA
- ORB's
- ORBacus
- C++ Client
- Tests
- Summary

C++ & Java Inter-operability



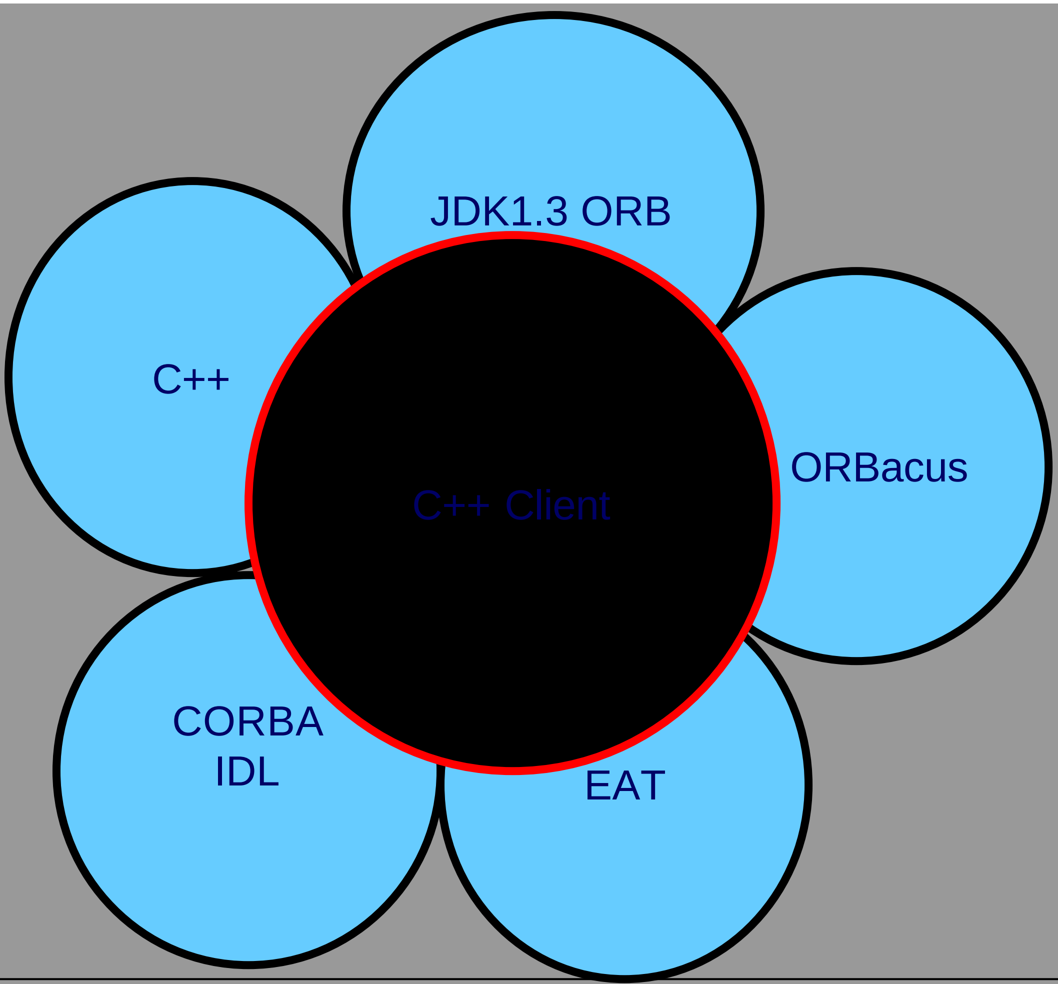
AQUILA

Proposed Solution



IIOP : Inter-ORB Protocol

Inter-operability



CORBA

- The Common Object Request Broker Architecture (CORBA) is the product of a consortium called Object Management Group (OMG)
- The notable exception is Microsoft, which has its own competing object broker called the Distributed Component Object Modeling (DCOM)
- The CORBA object bus defines the shape of the object bus that live within it and how they inter operate.
- CORBA is designed to allow intelligent components discover each other and interoperate on an object bus.

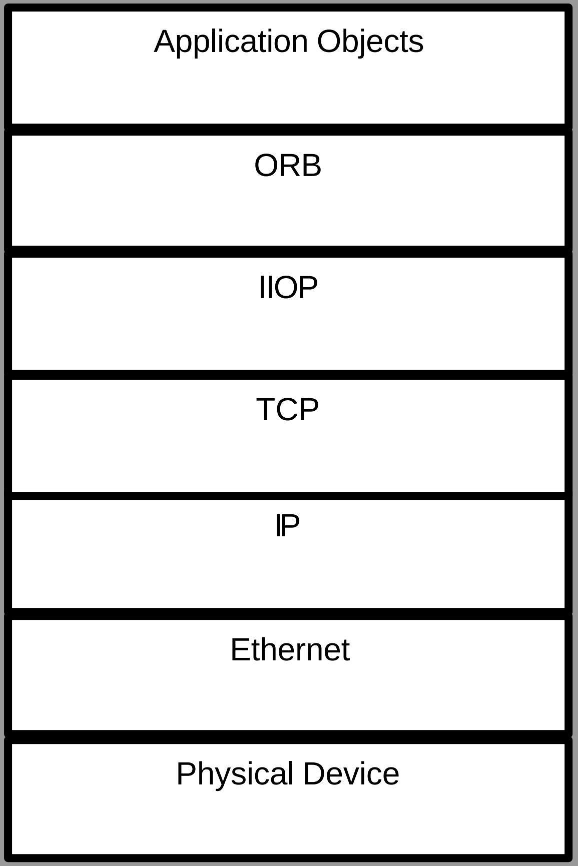
AQUILA

IDL : Interface Definition Language

- The CORBA IDL is purely declarative and provides no implementation details
- IDL can be used to define API's concisely and it can cover important issues like error handling
- IDL provides operating system and programming language independent interface to all the services and components that reside on the CORBA bus.
- It allows client and server objects written in different languages to interoperate

IIOP : Internet Inter-Orb Protocol

- With CORBA 1.0, the OMG left it up to the vendors to implement a protocol for their ORBs and objects to use over a network
- This allowed for interface operability through the use of IDL, but ORBs could not communicate with one another
- The CORBA 2.0 specification provides the methodology ORBs need to communicate General Inter-ORB Protocol (GIOP)
- IIOP maps GIOP messages to TCP/IP



C++ Supporting ORBs

- **TAO ORB**

  ***T**he **A**CE **O**rb*

  Developed by Washington University with support from Object Computing, Inc.

- **MICO ORB**

  ***M**ICO **I**s **CO**RBA*

  freely available and fully compliant implementation of the CORBA standard

- **ORBacus**

  Developed by IONA Technologies

  Apart from C++, also supports Java

AQUILA

JDK ORB

- Developed to enable objects to interact regardless the language used to code them
- The drawback of the ORB is that it does not support C++ coded objects
- The interoperability of the Java ORB with other vendor's ORBs have not been tested
- The only way to know if it can inter-operate with ORBacus is to develop a C++ ORB and test it.

ORBacus Guide

- Install the license key
- Install *nmake*
- Do necessary changes in *config\Make.rules.mak*
- Enable run-time library
- Run-Time Type Information
- Include library files that are needed to compile ORBacus applications

The C++ Client

```
Class SampleClient{
public: SampleClient {
  …
}
protected: void start(char* eatName){
 …
}
public: static void main(int argc, char *argv[]){
   …
}
};
```

The Modules

- public: static void main(int argc, char *argv[])

  Initializes an object of the class *SampleClient*

  Calls the *start* function of this class.

- public: SampleClient

  This is the constructor of the *SampleClient*

  Connects the client to the server using the Naming Service

- protected: void start(char* eatName)

  Interface of the client with the EAT

  Call the various services provided by the EAT like login, service request and application request.

Running With The Server

- Client interoperates with the server over the IIOP
- ORBacus encapsulates the requested object into IIOP
- Naming Service maps the object the server
- JDK unwraps the IIOP message and refers the object to the server
- JDK 1.3's access mechanism to Interoperable Naming Service is proprietary
- JDK 1.4 supports Interoperable Naming Service but the server has yet not been tested on JDK 1.4

A not so Elegant Solution

- **Procedure**

  Start the Naming Service

  Copy the Interoperable Object Reference (IOR)

  Paste the IOR in the Client code

  Compile and run the Client

- **Disadvantage**

  Practical only on a test bed

Summary

- **Achievements**

  Study of AQUILA

  Study of CORBA and IDL

  Study of ORBacus

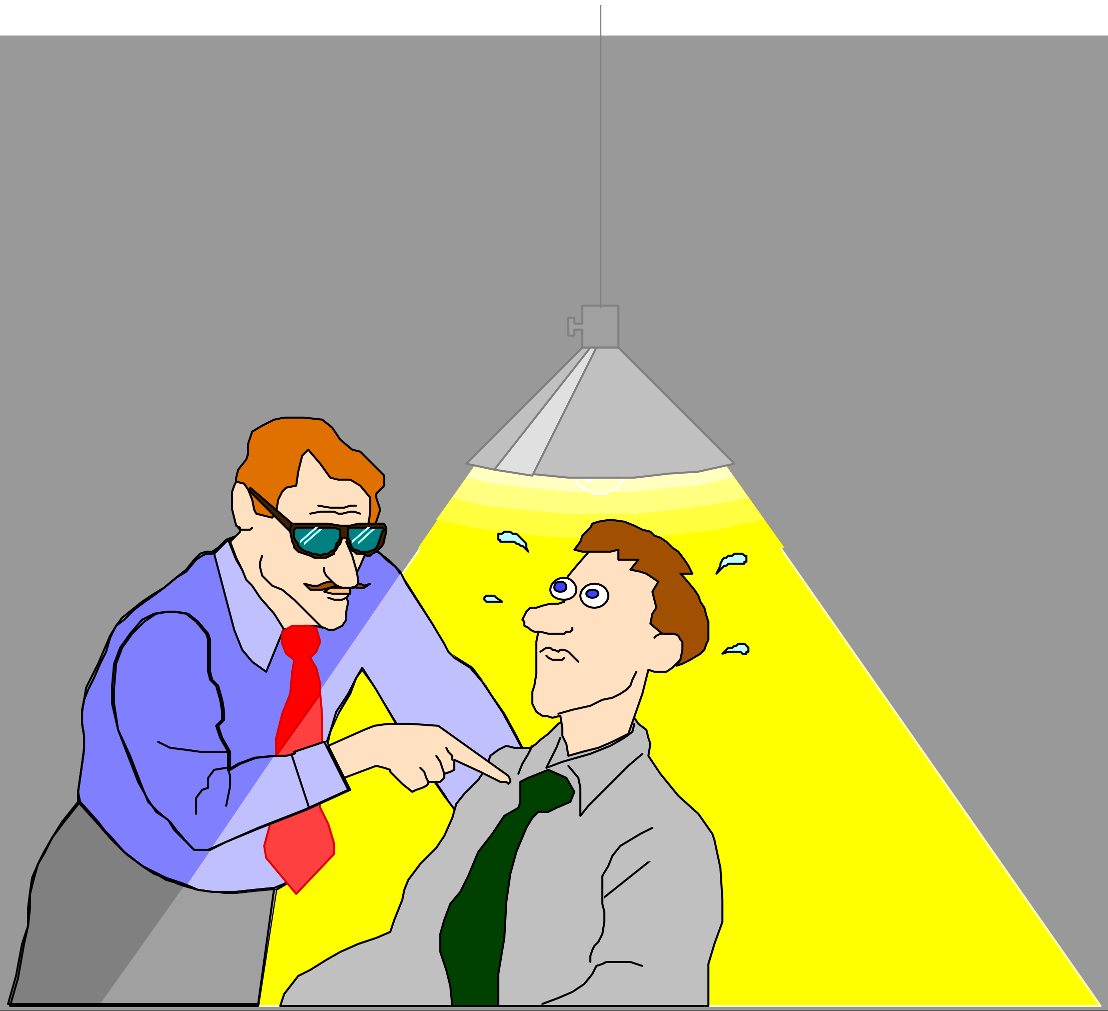
  Development of C++ Client

- **Learnt**

  QoS and AQUILA

  Inter-operability

Questions & Discussion

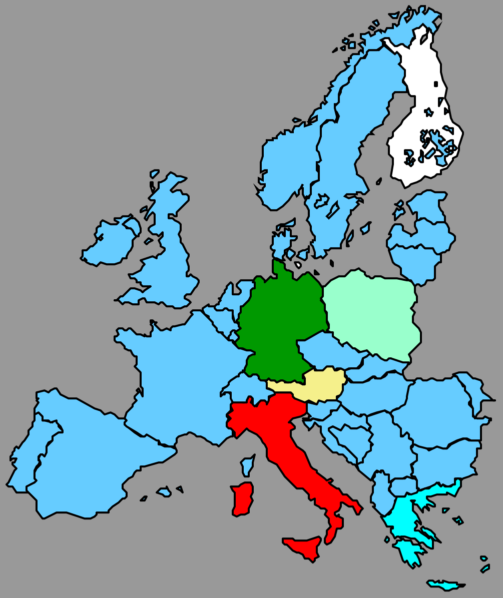


AQUILA

AQUILA (IST-1999-10077)

information society technologies

Adaptive Resource Control for QoS
Using an IP-based Layered Architecture



Thank you for your attention !

http://www-st.inf.tu-dresden.de/aquila/