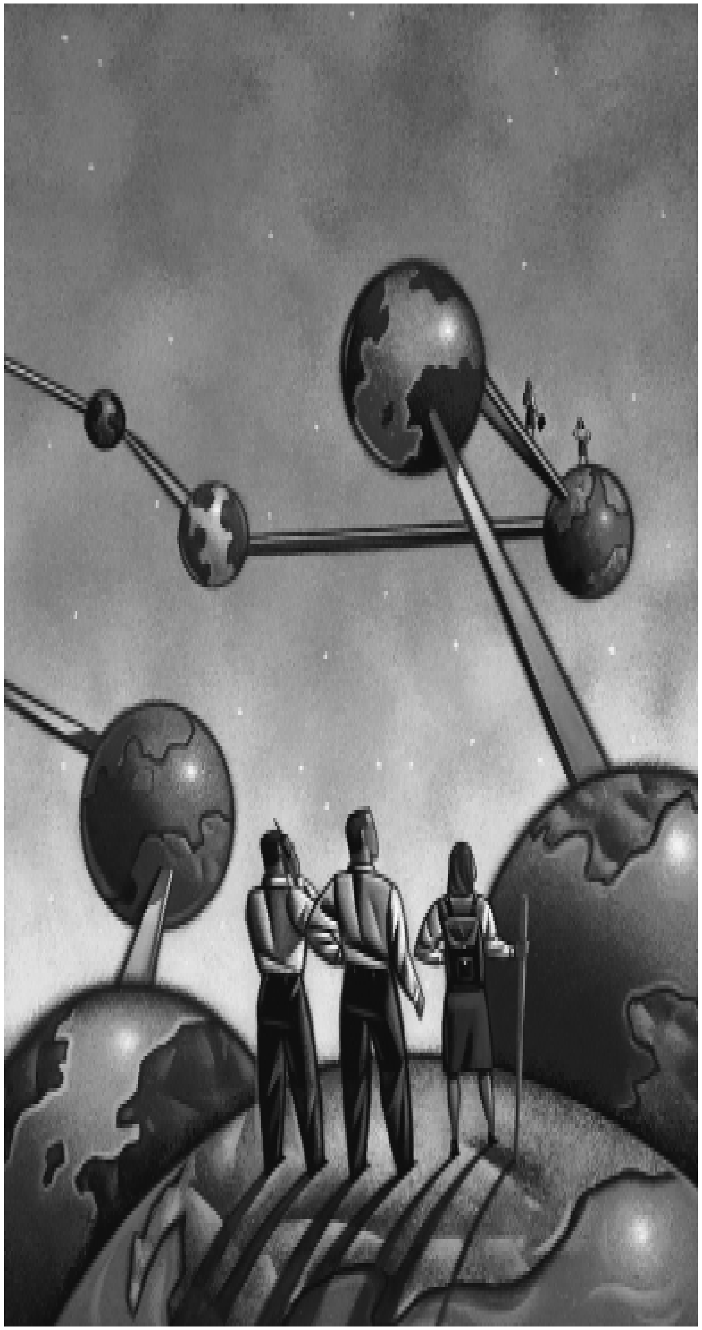
Dynamic *closed-loop* QoS enforcement - new frontiers in real-time CORBA

H. Hi Ph.D., A. Verma, S. Aslam-Mir Ph.D.
Implementation engineers – Vertel USA
www.vertel.com

vertel®



Synopsis

- Motivation for adaptive middleware
- CORBA evolution–
  - Technical drivers
  - Commercial drivers
- Adaptive RTCORBA systems
- Example drivers to RTCORBA evolution
- Adaptive approach
- Schematic of design

Motivation for new approach

- Current generation deployed RTCORBA systems are point solutions w.r.t. operational envelope
- Design for a limited number of operational points in performance envelope
- Inter-point transition done by extensive gain-scheduling approach
- This is acceptable by enlarge – leaky bucket approach
- Next wave of adopters are not willing to accept this
- Stringency and temporal correctness are more critical
- Researchers applying ‘reflexive adaptive’ techniques in middleware, NOT just at application level.

Motivation

- Mission critical embedded real-time systems must provide a reliable/fault-tolerant capability to provide some real-time service with a known level of QoS.
- <u>i.e the commanded QoS level must be maintained !</u>
- RT CORBA 1.0 provided us with the turbo-machinery to do this
- But the mechanisms to ensure maintained level of QoS has been application not middleware dependent – this is a problem.

Adaptive middleware

- Some will claim to be adaptive right now
- I will agree ☺ !
- BUT
- How does this help the end-user
- Need a mapping between the adaptation performance map of the ORB to that of the end-users QoS profile requirement or *contract*
- How will RT-CORBA help ?
- . . . . . Read on

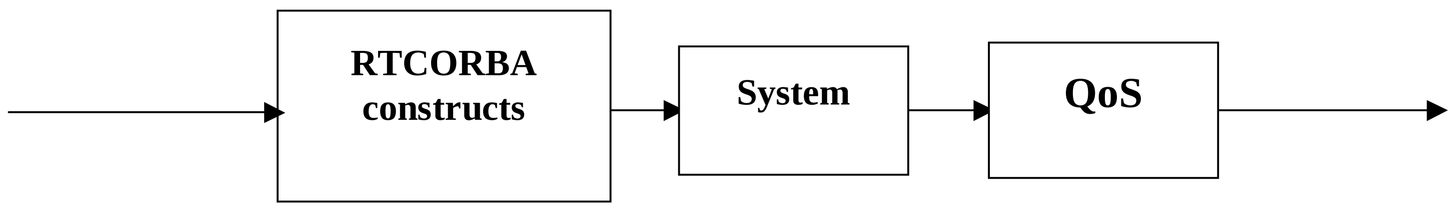
RT-CORBA evolution – a motivator

- A significant milestone in providing yet another canonical middleware architecture specification for distributed systems.
- Still based on simple interoperable protocol GIOP !
- Real-time and not so much embedded system focus
- Telecom was one of the earliest adopters BUT is in management plane of Access and heavyweight equipment vendor space – Lucent, Nortel Class 5 switch management systems . . .
- Large body of literature exists on this type of application of RTCORBA – Tellium Aurora Optical switch <u>beginning to allow RTCORBA permeation in the control plane.</u>
- Almost no body of literature approached CORBA for MEMs until recently – Rofriguez &Gill, Kopetz

RT-CORBA evolution – a motivator

- As we see an evolution of RT CORBA and refinements being made by implementers to achieve ever higher performance and real-time correctness
- WHY – squeeze more out of ORB-Tranport-RTOS n-tuple
- one trend becomes clear !
- RT middleware architecture specification needs to have an adaptive piece
- BUT – heres the good news -
- We see in the structure RTCORBA 1.0 and 2.0 a very important set of structures that allow for an open adaptive solution

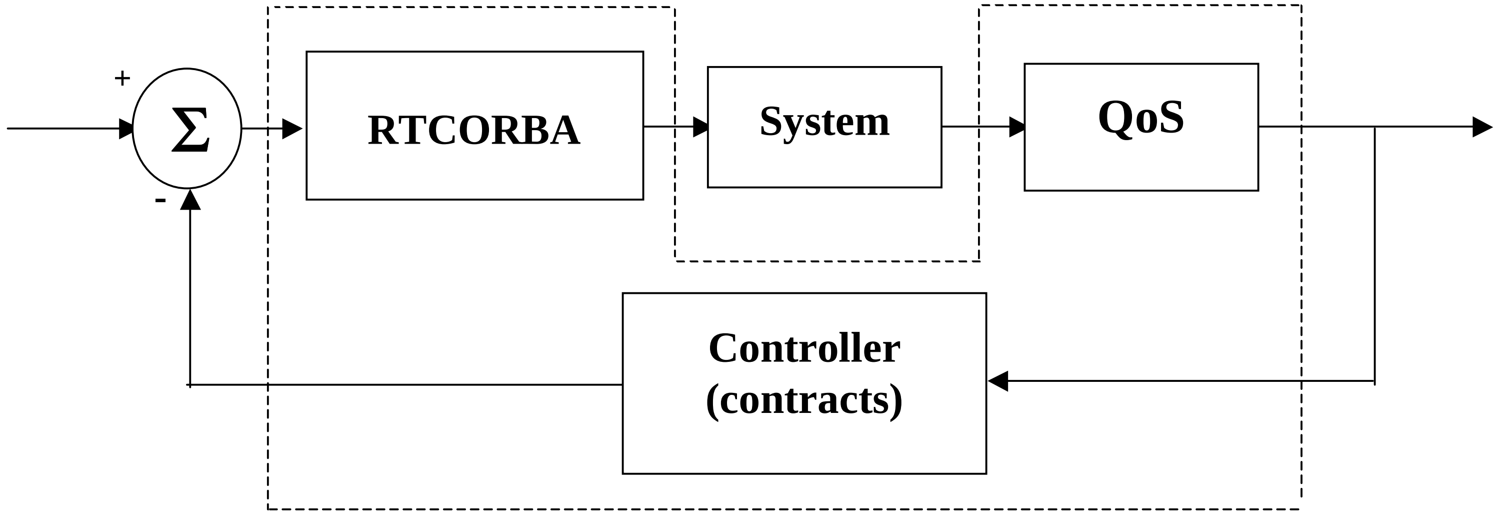
Current generation RT-CORBA



Open-loop system. Absence of feedback loop with a-priori design

2nd generation CORBA, *real-time capable middleware*

Adaptive RT-CORBA paradigm



Intelligent middleware – self-regulating

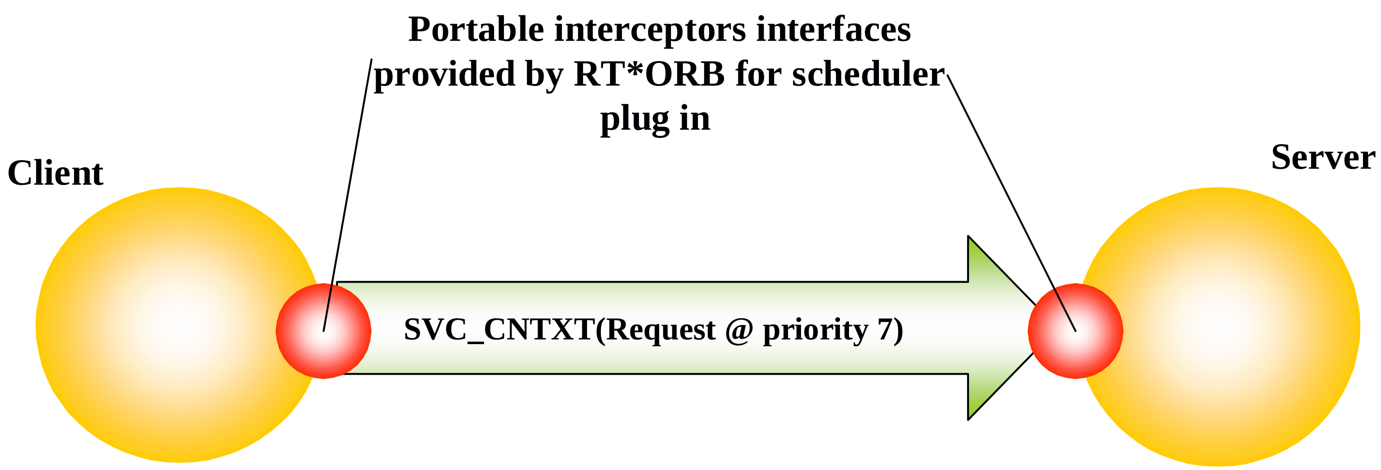
3rd generation *truly reflexive real-time middleware*

Dynamic closed-loop model – the good news

- CORBA::Policy
- CORBA::PortableInterceptors
- Real-time CORBA 1.0 specification priority *machinery*
- Real-time CORBA 2.0 distributable thread *machinery*
- RTCORBA Thread-pools and Priority banded connections
- Optional compliance point – Fixed priority scheduler
- Service Contexts
- CORBA Messaging constructs overlap RTCORBA – good

Examples of adaptive ORB mechanisms to control QoS

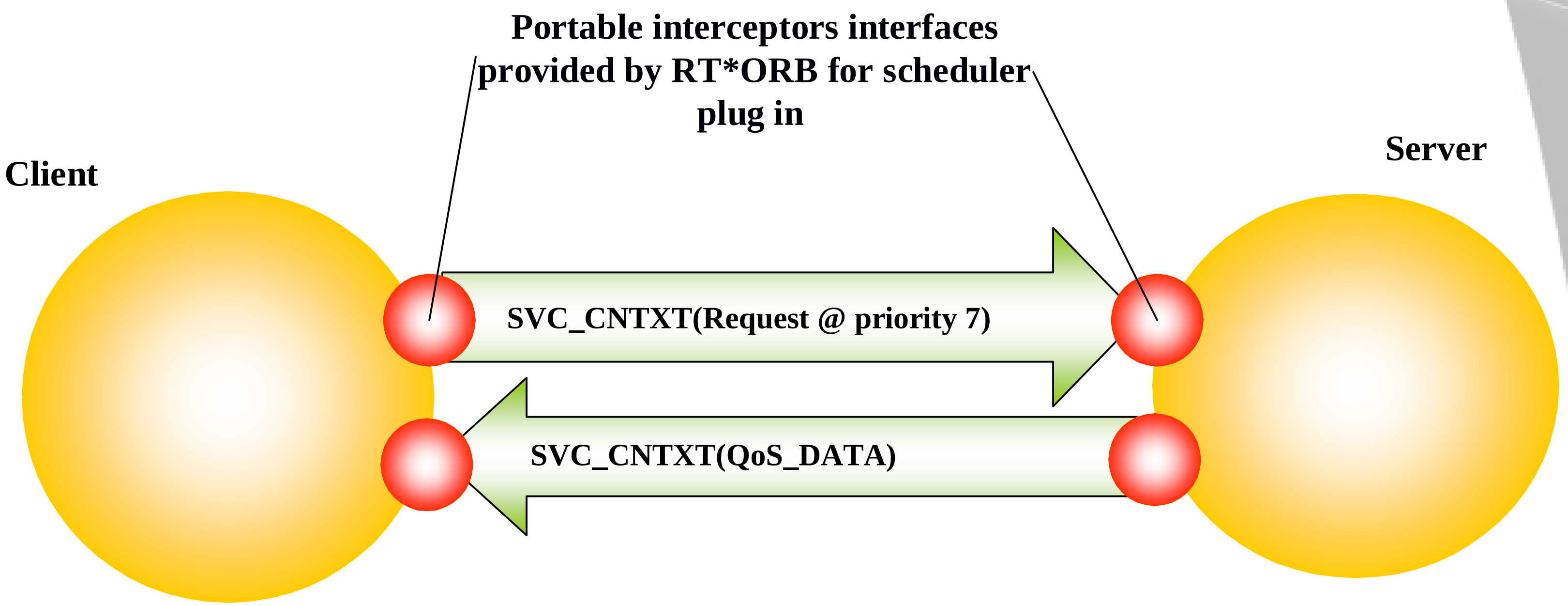
Client-propagated priorities



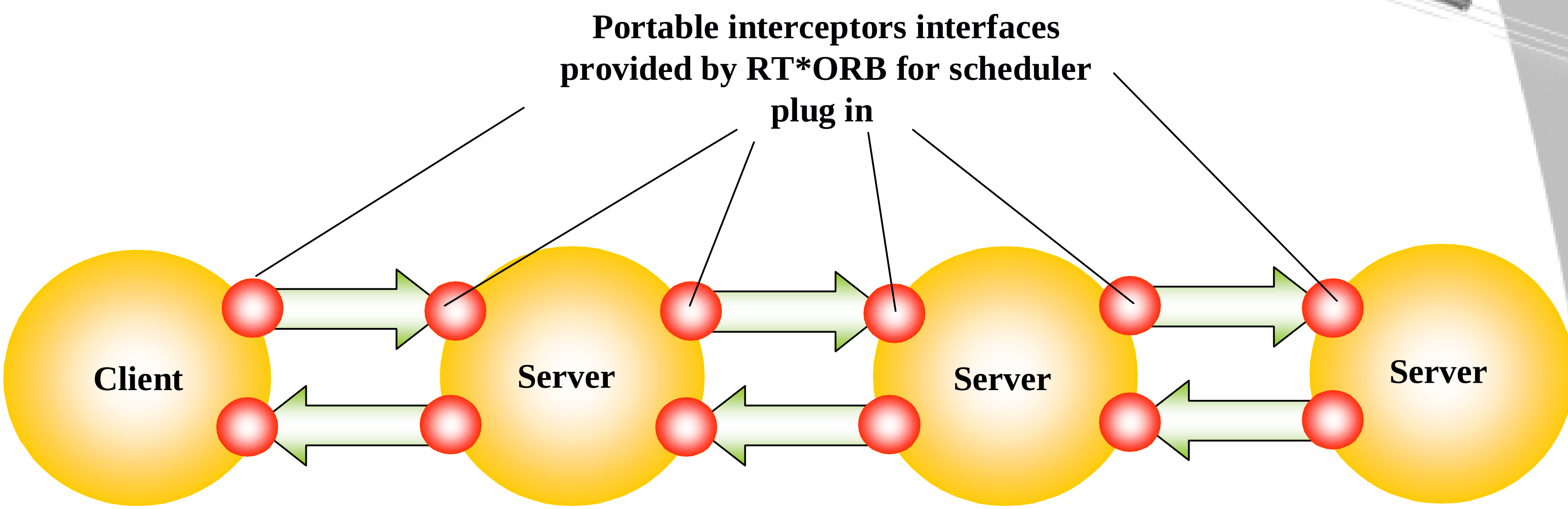
Client Propagated Model:

- **Can use Portable Interceptors (PI) for propagating priorities in service contexts.**
- **PI also enable Dynamic and Static scheduler hook-in**
- **PI additionally allow user to define their own QoS through client propagated model**
- **RT implementation should supply default client propagated model without having to use PI interfaces.**

Client-propagated priorities with crude feedback

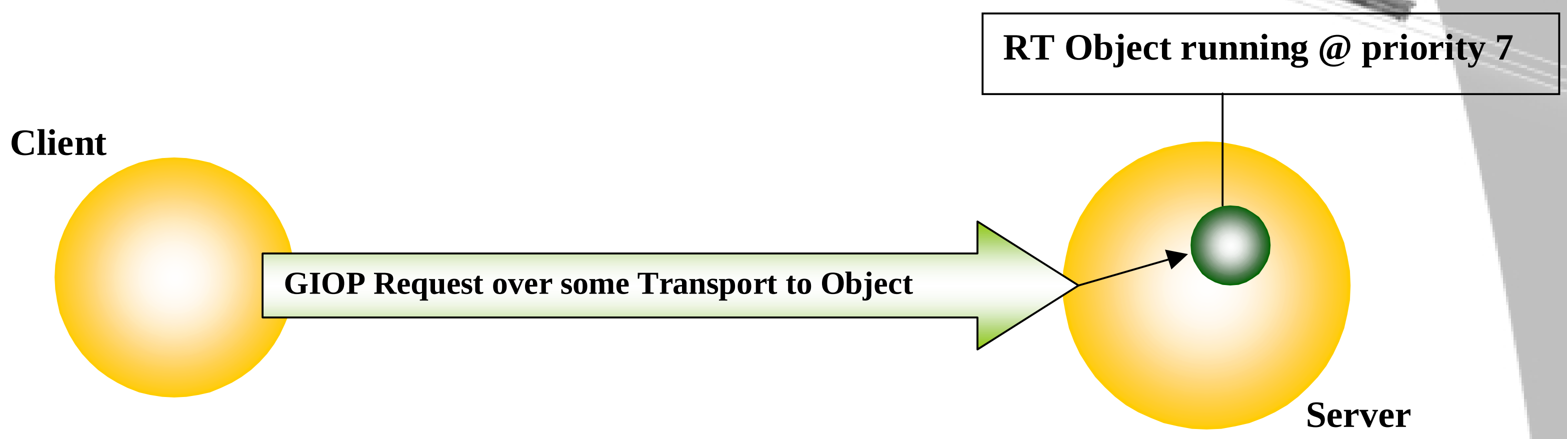


Client-propagated Multi-hop !



Use of Service Contexts and QoS_Data tuples for transmission in multi-hop calls for adaptive QoS control

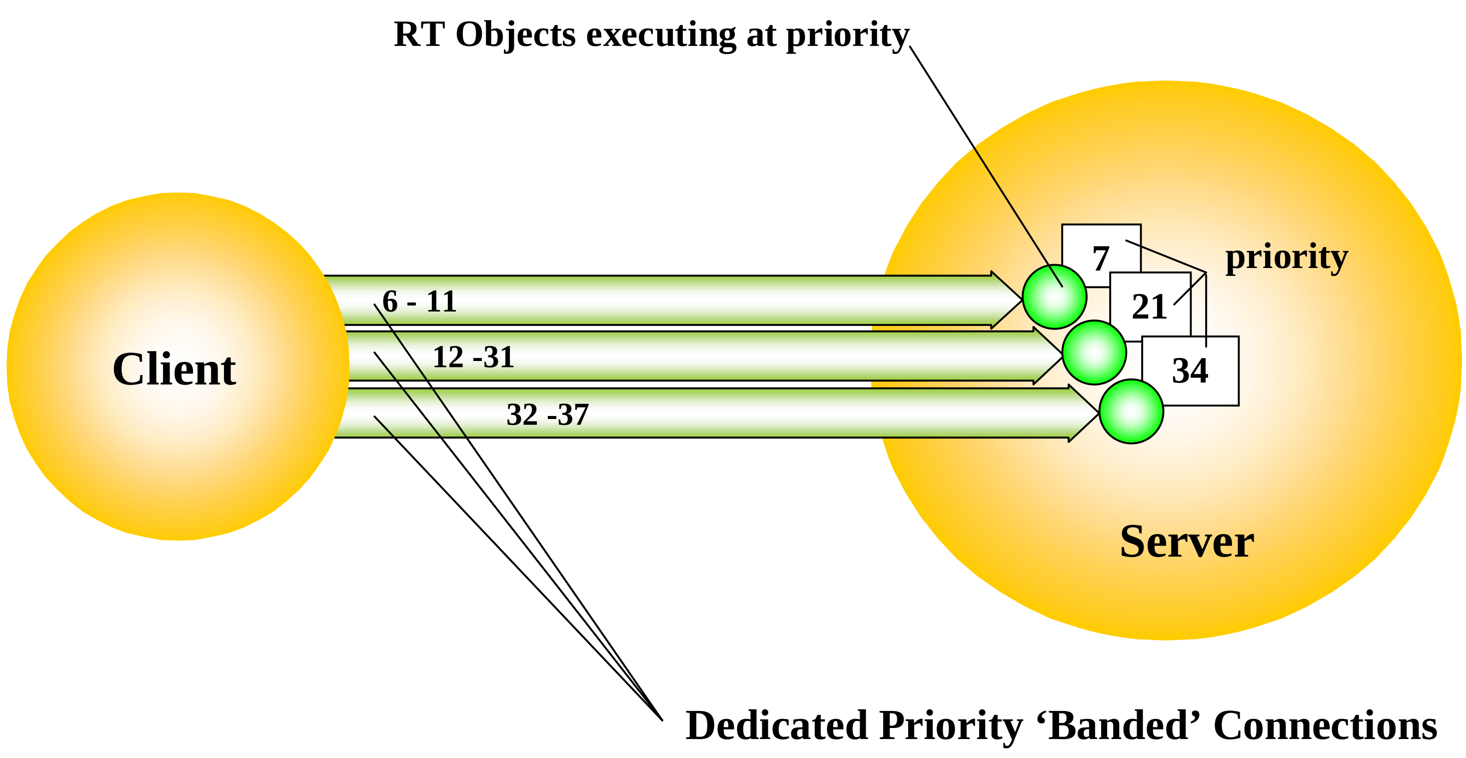
Server-declared priorities



<u>Server Declared Model:</u>

- **Can also be scheduled using dynamic and Static scheduler hook-in**
- **Scheduler enables user to impose own server side scheduling policies.**
- **e*ORB RT Edition supplies pluggable PI interfaces use also with Server declared model for interoperability with non real-time ORBs..**

Priority Banded Connections



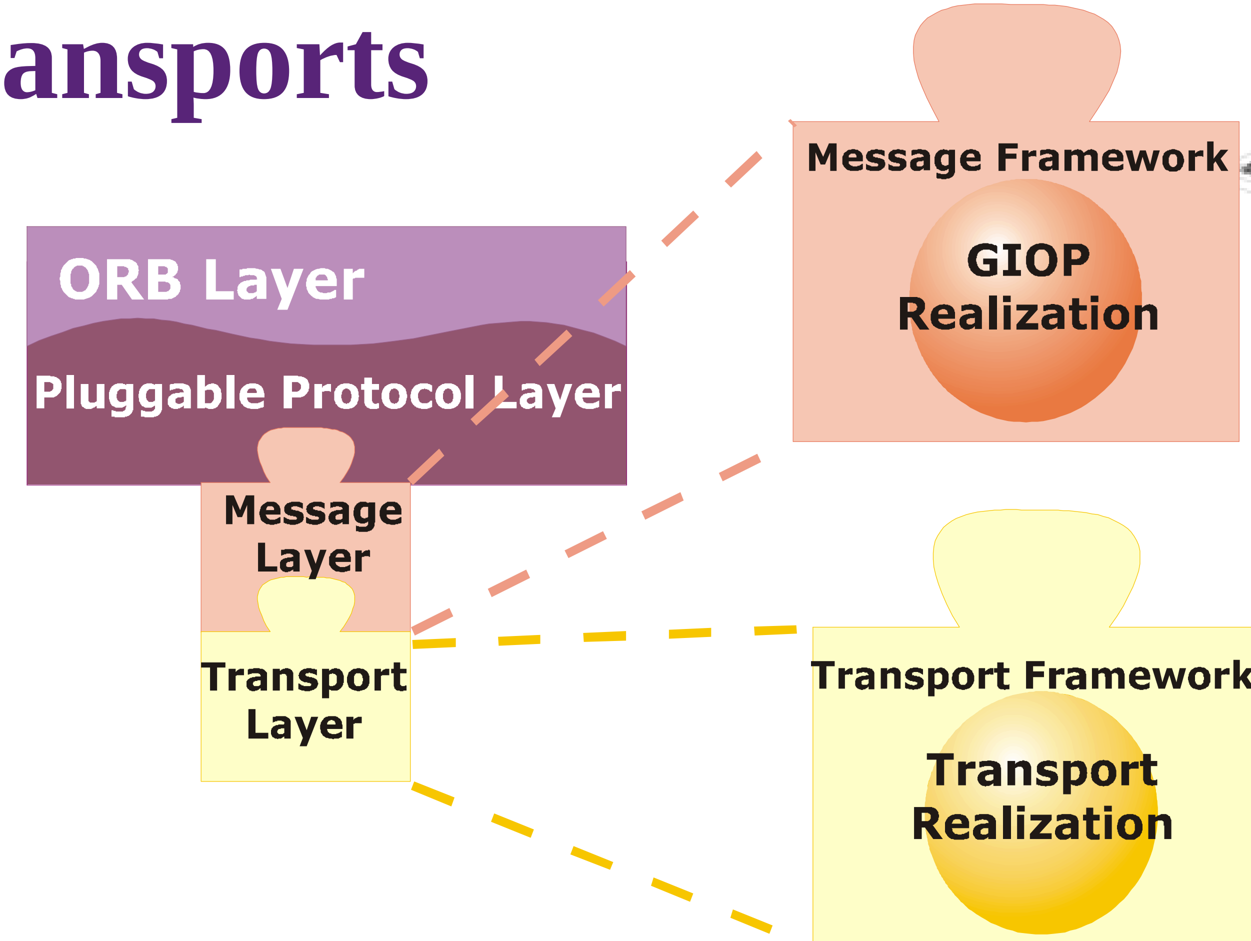
Dynamic closed-loop model – the bad news

- Footprint conflicts
- Loss of some performance possible
- Feedback loops to adjust ORB internals means to squeeze better performance has implications on internal design
- New ideas in adaptation with protocol drivers and smart memory allocation complicate the picture BUT
- Gains in laboratory experiments have shown significant advantages
- Manage cost of development and lifecycle
- Manage complexity of the resulting software
- - enter QoS components !!!!!

Technical drivers

- Research into distilling down CORBA
- Next generation CORBA – minimum
  - reduce memory footprint several orders
  - increase speed, thruput
  - drastically reduce code size
- Could now plug-in new and different transports – PCI, ATM, VME, RACEWay so . . .
- Adaptive ORB should make decisions about which connections and transports to use based on internal heuristic machinery

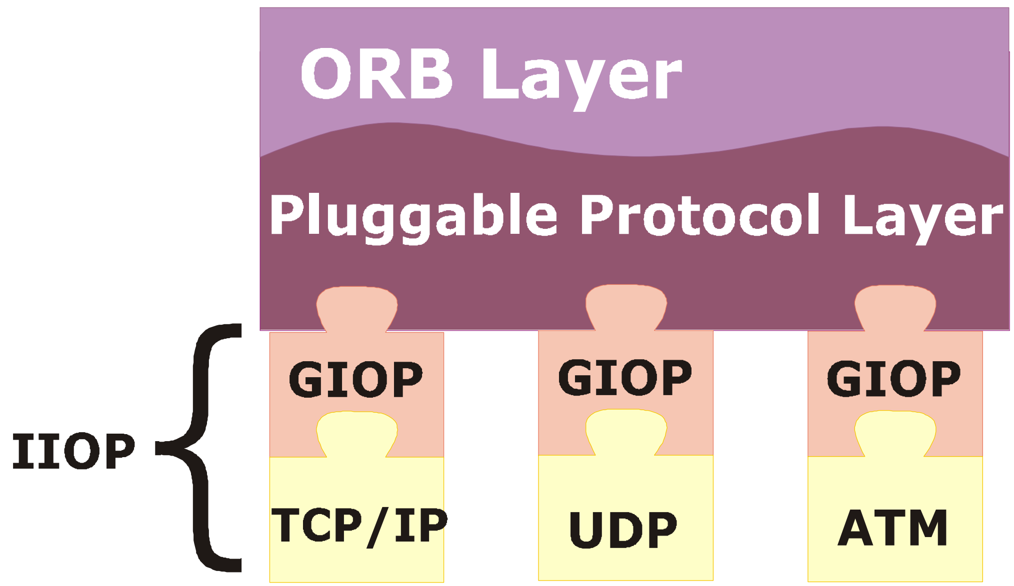
CORBA and Pluggable Transports



- ORB can run over multiple transports simultaneously
  - e.g., wireless TCP, WAP, mobile-IP R/U UDP, SSL, ATM, cPCI,
  - widely different communications characteristics

ORBs are becomg Transport-neutral

- Transport Protocol stacks dynamically selected
- Multiple stacks simultaneously
- Transparent to application logic



Commercial drivers

- Allure of COTS approach to build h/w and s/w started to attract top end embbeded real-time technology consumers, BUT
- CORBA in management plane only, not in control plane, or very limited capability
- Softswitch application types of latencies are sought – especially those experienced and sustained at peak sporadic load times.
- Can RT CORBA meet that challenge ?

Embedded real-time (RT) CORBA systems

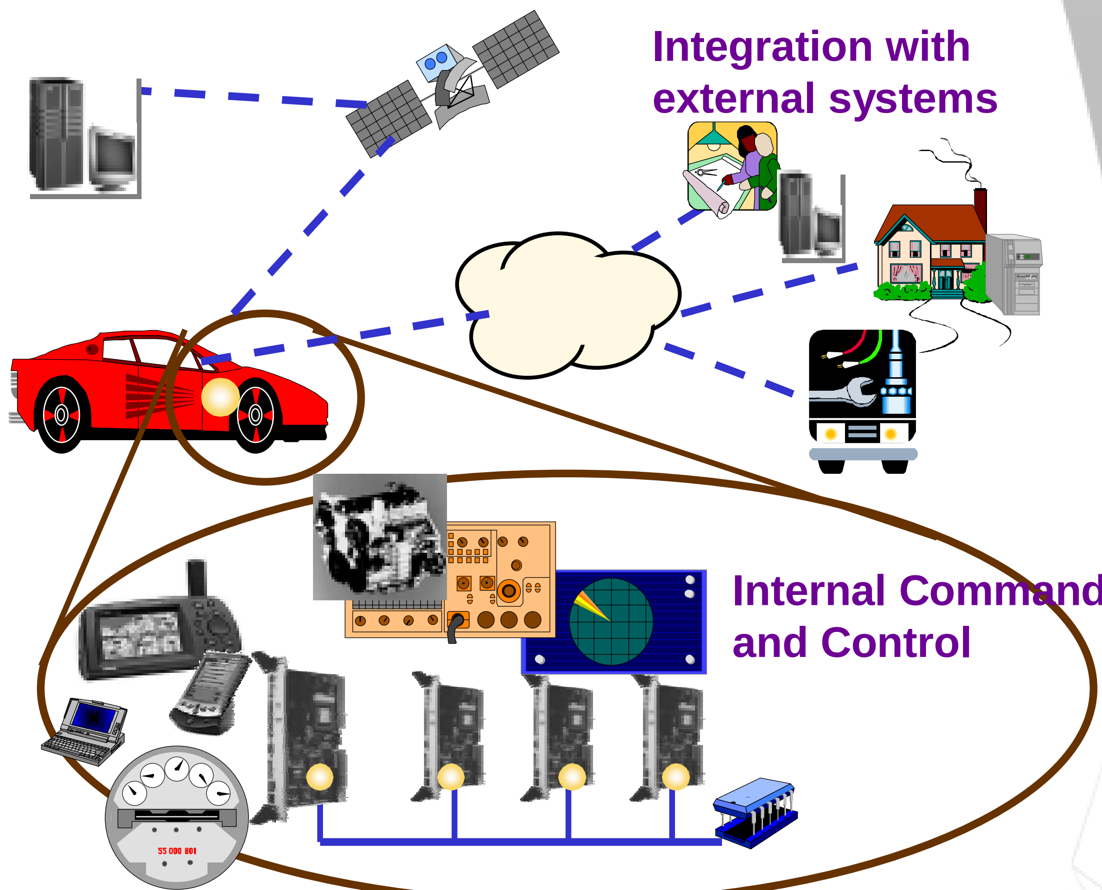
- First approach was specification route –
- Minimum CORBA Spec.
- Fails to address several major issues that are of concern in developing embedded systems.
- Real-time systems addressed separately in real-time CORBA Specification
- Uptake has been a mixture of full, minimum and real-time CORBA, with difficult demarcations.
- Most uses are a hybrid mixture of all
- Central focus is QoS achievement – not complete
- **Open loop solution**

Mixture-Grade CORBA

- Focus here is some QoS with fault-tolerance (FT).
- Many industries have taken a mixed approach
- Adoption of minimum CORBA ‘inside-the-box’
- Adoption of some forms of real-time CORBA to small localized microcosms inside the box or in wireless segments
- Adoption of enterprise CORBA in service layer apps, augmented with FT infrastructure.
- Mixing of CORBA services e.g. notification, naming, fault-tolerance etc,
- Management functions link minimum and full enterprise CORBA legs

Telematics
- Advanced Communications

- Command & Control
- Auto PC environment
- Integration with advanced enterprise services



Adaptive approach

- Many research units and industry have moved on to try to achive a closed loop solution -
- This solution can give better temporal performance
- Adaptive in key

….. So …..

Define an adaptive model

- QoS centric with fault-tolerance – eg assume call can be delivered under some {set} of circumstances
- *Mathematically modeled* (Empirical and deterministic)
- Model of dynamic memory usage – lowest footprint
- Protocol plug-in enhancements – black art
- Able to leverage esoteric features of exotic transports
- Effective RT scheduling
- Light-weight bootstrap and discovery services
- Built in minimal fault-tolerance result in
- **Closed loop QoS enforcement i.e. self-regulating**

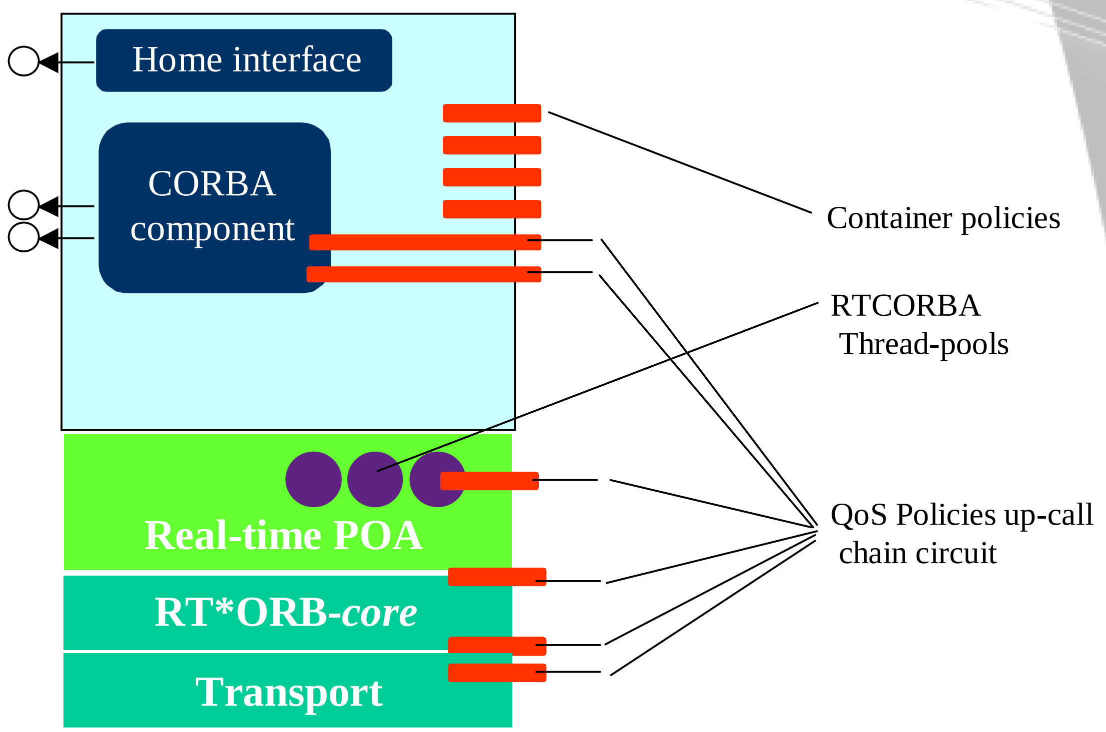
Examples – ORB can make some decisions based on policies stipulated

- ➢Co-location stubs
- ➢Shared memory optimizers
- ➢Internal ORB QoS interfaces invocable through interceptors
- ➢Dynamically linked/loaded servants
- ➢Dynamic (re)configuration & upgrade****
- ➢QoS adjusting machinery e.g.load balancing, netwrok reservation
- ➢Dynamic transport selection.
- ➢Feedback based control of jitter and latency data into the call chain
- ➢Feedback based regulation of call timing profile (not the same as minimizing the latency and bounding the jitter)

But now theres a Problem -

- We were trying to achieve good QoS characteristics
- We achieved greater complexity in programming model
- Increased cost of ORB and development and deployment cycle
- **Solution : use a stripped down abstraction of a <u>CORBA Component</u> to try and control the complexity through the <u>container</u> model**
- **Now we have a tradeoff**

What does this ‘*simple*’ design look like



Thank-you