

The Design and Performance of a Pluggable Protocols Framework for Real-time Distributed Object Computing Middleware

Carlos O’Ryan, Fred Kuhns, Douglas C. Schmidt, Ossama Othman and Jeff Parsons

coryan@uci.edu
<http://www.ece.uci.edu/~coryan/>

Motivation: Rapid Growth in Hand-held Devices



Lewis:98 (IEEE Computer Jan'98)

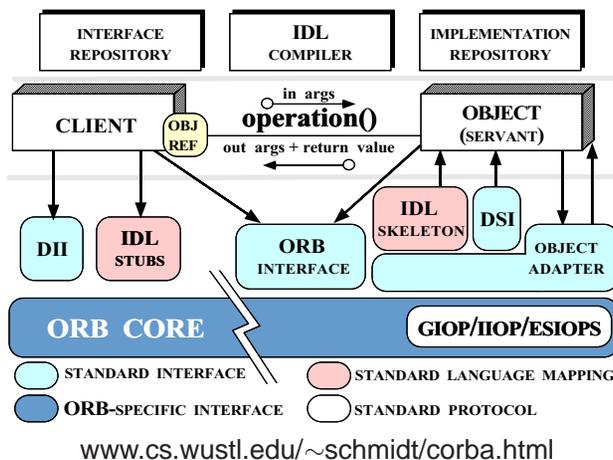


Comerford:98 (IEEE Spectrum May'98)

- **Devices** → PDAs, WebPhones, WebTVs, handsets
- **Requirements** → Low power consumption, small memory/message footprint, efficient/predictable performance



Candidate Solution: CORBA

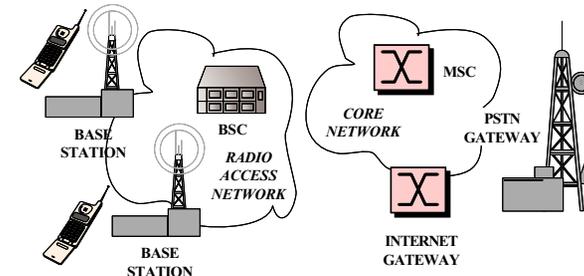


Goals of CORBA

- Simplify distribution by automating
 - Object location & activation
 - Parameter marshaling
 - Demultiplexing
 - Error handling
- Provide foundation for higher-level services



Caveat: Limitations of CORBA for Wireless



Requirements

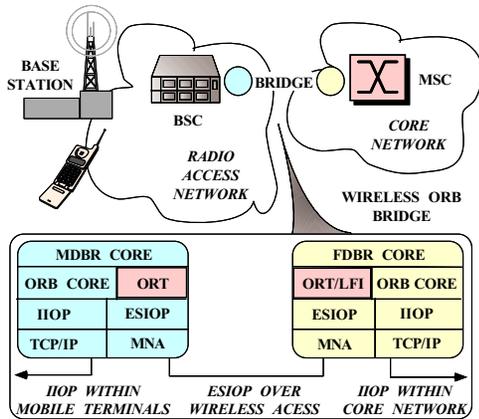
- **Location transparency**
- **QoS transparency**
- **Minimal footprints**

Limitations

- **Lack of appropriate protocols**
- **Lack of real-time and async QoS features**
- **Lack of performance optimizations**



OMG Approach: Wireless CORBA



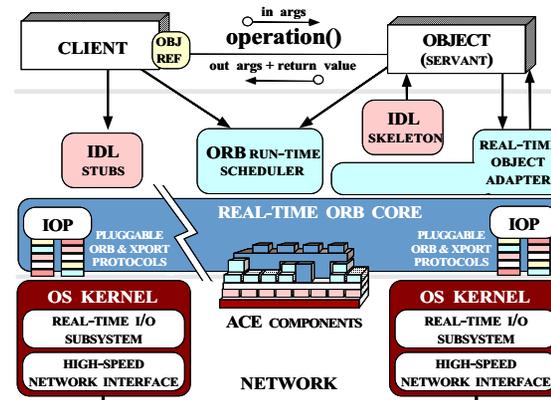
Overview →

- Mobility domain and fixed domain bridges handle mobility
 - e.g., relocatable object references via location register
- Lightweight ESIOP reduces message footprint

www.omg.org/techprocess/meetings/schedule/Wireless_Access_&_Control_RFI.html



Our Approach: The ACE ORB (TAO)



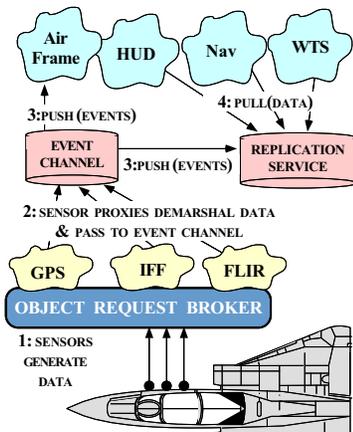
TAO Overview →

- 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

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



Motivation



- Middleware should exploit the good features of its environment
 - The “Middleware Hypocratic Oath”
- CORBA is increasingly being used in distributed embedded systems.
- Protocol selection must be done “as late as possible” in the design cycle



Lewis:98 (IEEE Computer Jan'98)

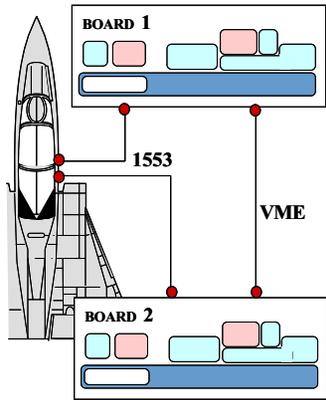


Comerford:98 (IEEE Spectrum May'98)

- **Devices** → PDAs, WebPhones, WebTVs, handsets
- **Requirements** → Low power consumption, small memory/message footprint, efficient/predictable performance



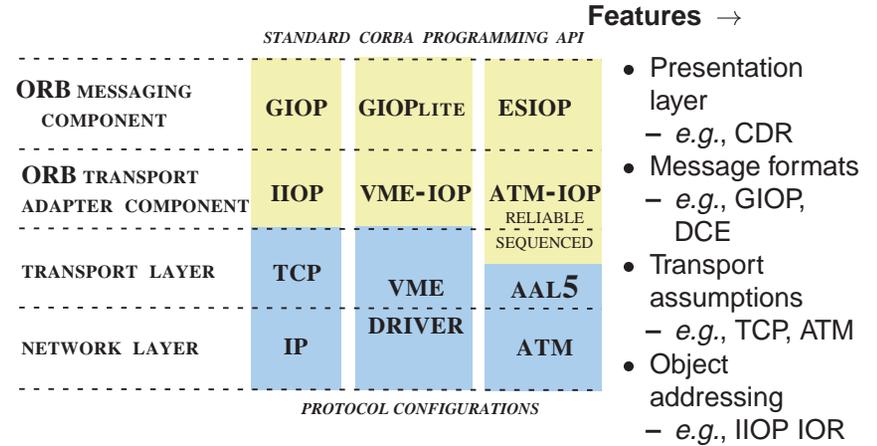
Problem: Hard-coded ORB and Transport Protocols



- GIOP/IIOP are not sufficient for all environments, e.g.:
 - GIOP messages may be too large
 - TCP lacks necessary QoS
 - Legacy commitments to certain protocols
- Existing ORBs don’t support “pluggable protocols”
 - Both *ORB messaging* and *transport* protocols must be pluggable



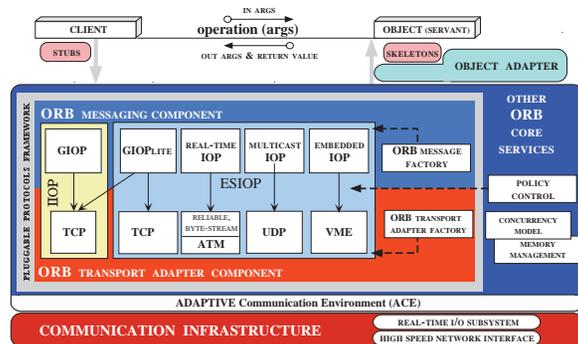
Context: CORBA Protocol Architecture



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



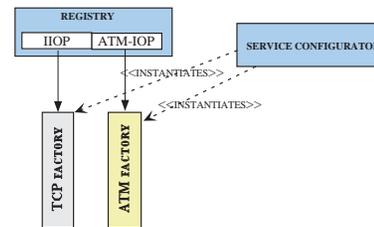
Solution: TAO’s Pluggable Protocols Framework



- Pluggable **ORB messaging** and **transport** protocols
- Highly efficient and predictable static and/or dynamic configuration



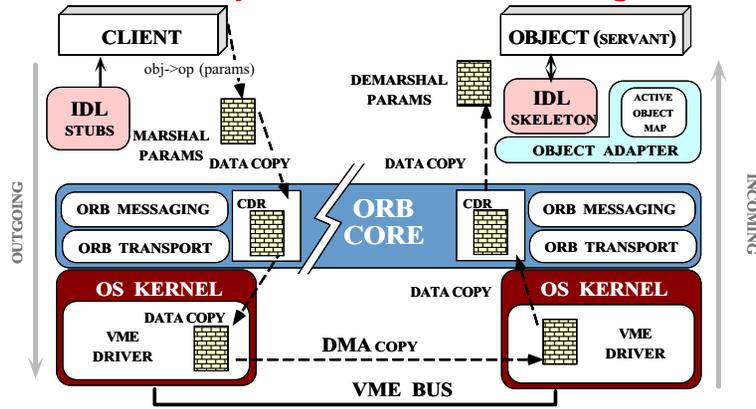
Design Challenge: Adding New Protocols Dynamically



- **Problem:** the range of protocols available can be modified late in the development cycle.
- **Solution:** Use the Service Configurator pattern to:
 - Dynamically load the registry class
 - or, dynamically load a registry class that dynamically loads its contents



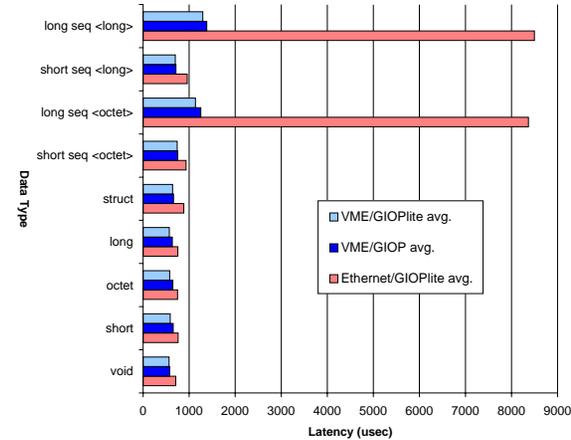
Embedded System Benchmark Configuration



VxWorks running on 200 Mhz PowerPC over 320 Mbps VME & 10 Mbps Ethernet



Ethernet & VME Two-way Latency Results

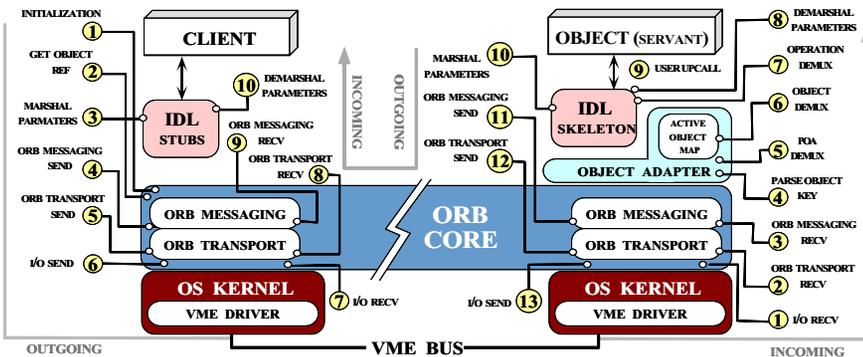


Synopsis of Results

- VME protocol is much faster than Ethernet
- No application changes are required to support VME



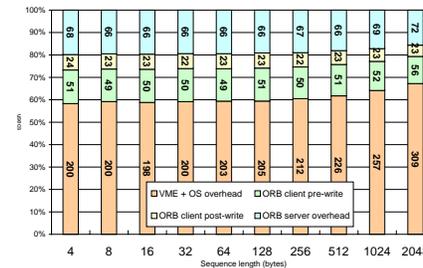
Pinpointing ORB Overhead with VMEtro Timeprobes



- Timeprobes use VMEtro monitor, which measures end-to-end time
- Timeprobe overhead is minimal, i.e., 1 μsec



ORB & VME One-way Overhead Results

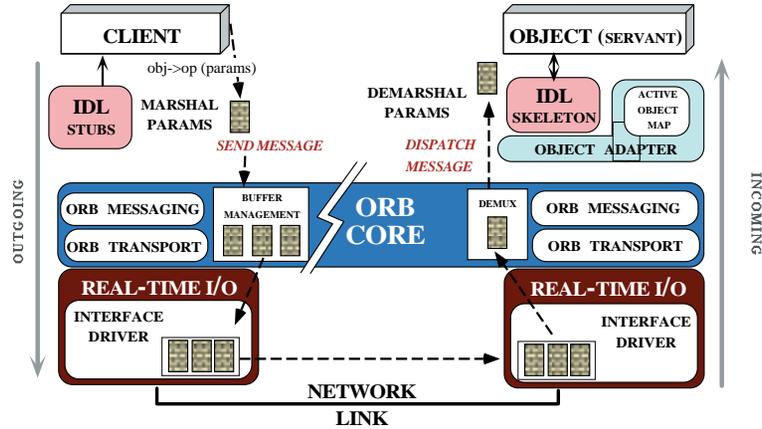


Synopsis of Results

- ORB overhead is relatively constant and low
- e.g., ~110 μsecs per end-to-end operation
- Bottleneck is VME driver and OS, not ORB



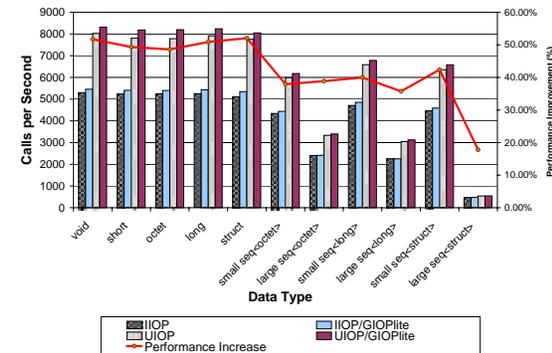
Workstation Benchmark Configuration



Debian Linux running on 400 Mhz workstation over Local IPC



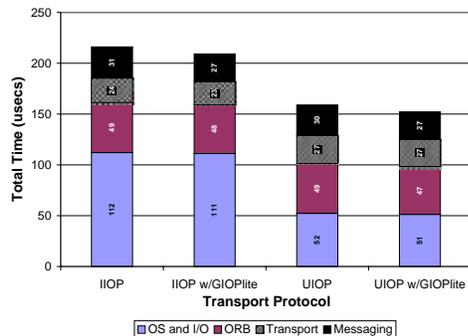
Loopback vs. Local IPC Results



- Significant improvements when using Local IPC for all cases
- The effect of message footprint is small
 - But would be more significant for low bandwidth links



ORB & Transport Overhead Results



Synopsis of Results

- ORB overhead is relatively constant and low
 - e.g., ~49 μsecs per two-way operation
- Bottleneck is OS and I/O operation



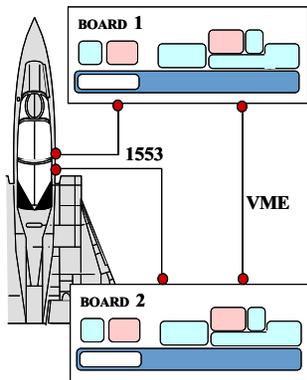
Open Issues for Wireless CORBA

	STANDARD CORBA PROGRAMMING API		
ORB MESSAGING COMPONENT	GIOP	GIOPLite	ESIOP
ORB TRANSPORT ADAPTER COMPONENT	IIOp	VME-IOP	ATM-IOP RELIABLE SEQUENCED
TRANSPORT LAYER	TCP	VME DRIVER	AAL5
NETWORK LAYER	IP		ATM

- Where should terminal mobility be supported?
 - i.e., in Request Bridge vs. Mobile IP
- What types of ESIOPs are necessary?
 - e.g., IIOp variants or entirely new ORB messaging protocols
- How will CORBA services be affected?



Concluding Remarks



- Pluggable protocols are essential to meet the needs of many CORBA applications, such as wireless and embedded systems
- Currently there are several efforts in the OMG to standardize pluggable transport and signaling protocols
- The Portable Interceptors specification will define standard interfaces for serialization

Web URLs for Additional Information

- **These slides:**
~schmidt/TAO/pluggable_protocols4.ps.gz
- **Information on pluggable protocols:**
~schmidt/pluggable_protocols.ps.gz
- **Information on CORBA:**
~schmidt/corba.html
- **Information on ACE:**
~schmidt/ACE.html
- **Information on TAO:**
~schmidt/TAO.html