MicroQoSCORBA: and Related Projects

Dr. David Bakken
Prof. Anneliese Andrews
Professor Carl Hauser
Dr. Kathleen Warren

WASHINGTON STATE UNIVERSITY

School of Electrical Engineering and Computer Science

Schedule

0930 Overview (Bakken)

0945 RealTime Issues (Hauser)

1000 More MicroQoSCORBA Update

1010 Overview of SWEng Research: testing, validation, etc.

1040 Discussion, Q&A

We want to learn about your related issues, problem domains, etc.

1115 Adjourn

Summary

- MicroQoSCORBA the only framework that
  - Is a “bottom-up” rethinking from the device level of what should be configurable, and in what ways
  - Is tailorable for a given application to the wide range of
    - Device constraints, ***and***
    - Application-dictated constraints
    - ***with*** a fine granularity of configuration constraints
  - Supports both “functional” and QoS properties
    - Security and fault tolerance as well as real-time performance are all key QoS properties
    - Highly extensible: easy to add more QoS mechanisms

MicroQoSCORBA

- CORBA-compliant ORB, interoperates with TAO, COTS …
- Baseline functionality configurable: **strip out unneeded code**
  - Node **roles**: Control flow, data flow, interaction style
  - **SW I/O**: data representation, protocols, direct/indirect
  - **IDL Subsetting**: GIOP message types, param types, data types, exceptions, message payload
- Quality of Service (QoS): Rich subsystems (details next page) for
  - Fault Tolerance
  - Security
  - Intrusion Detection (April 2004)
  - Bluetooth and wireless adaptation (April 2004)
- Java and C++ versions (Not all config choices on both languages)
- Primitive CASE tools allow configuration of above choices

MicroQOSCORBA QoS Subsystems

- QoS subsystems configurable: **choose what is needed**
- **Fault Tolerance**:
  - **Redundancy**: temporal, spatial, value
  - Message **reliability**: best effort, reliable, atomic, uniform
  - Message **ordering**: FIFO, Causal, Total
- **Security**
  - **Confidentiality** via encryption: AES, DES, Rot13
  - **Integrity**: Message digests (>10), ciphers (>10), message authentication codes
  - Availability: see Fault Tolerance above
  - **Accountability**: shared secrets

News Flash!!

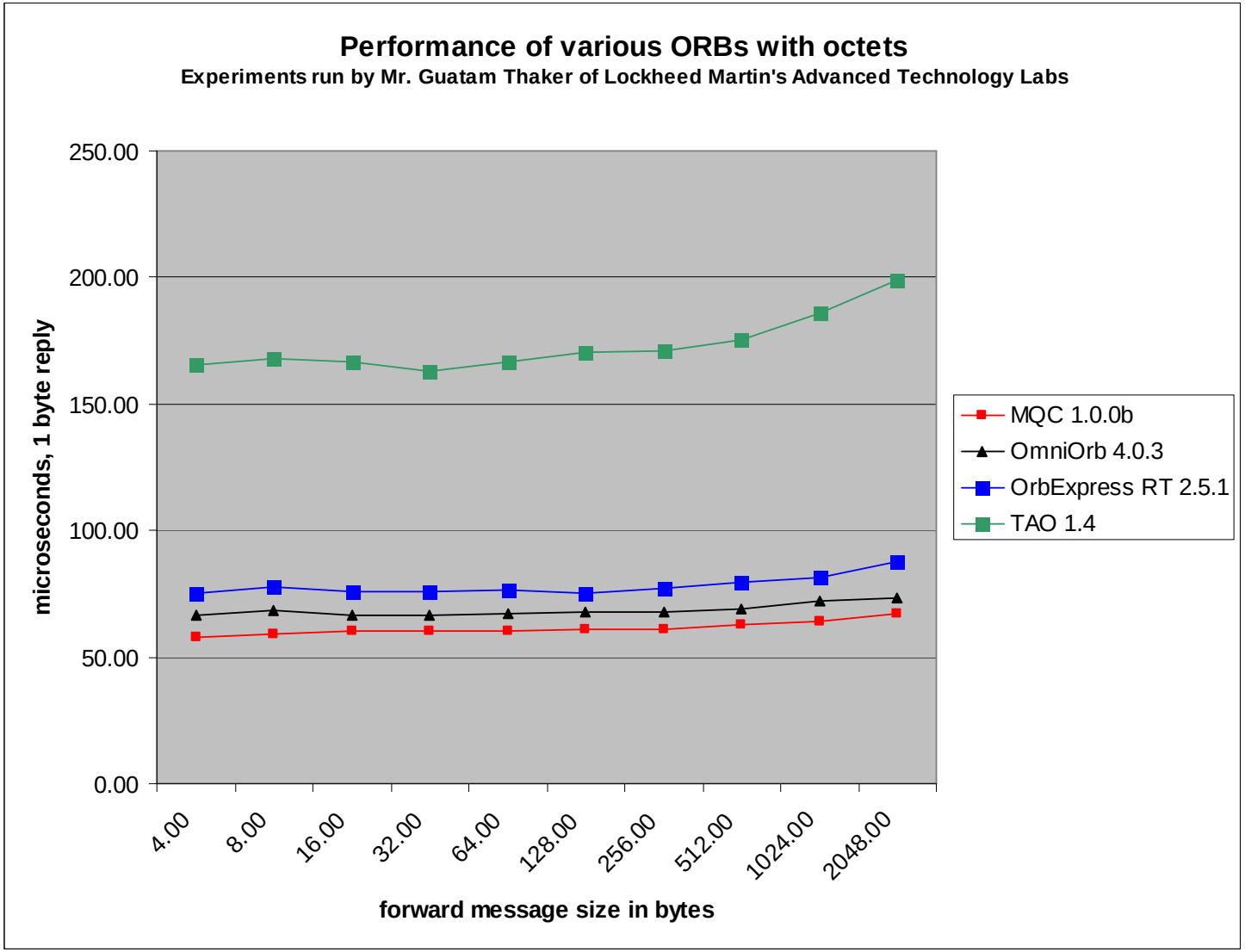
- April 4, 2004 update:
  - MicroQoSCORBA's C++ beta version is measured as the fastest middleware (academic or commercial) by Mr. Guatam Thaker, who for many years has benchmarked a variety of middleware systems for Lockheed Martin. A few graphs follow….
  - For more details see the "MW Comparator" at:

    http://www.atl.external.lmco.com/projects/QoS/

    (you can custom generate a lot of graphs!!)
  - Note: some of our excel graphs have been truncated at message size 2048, both to zoom in on the relative costs but also because embedded systems (or any non-multimedia app) rarely need to send bigger messages.

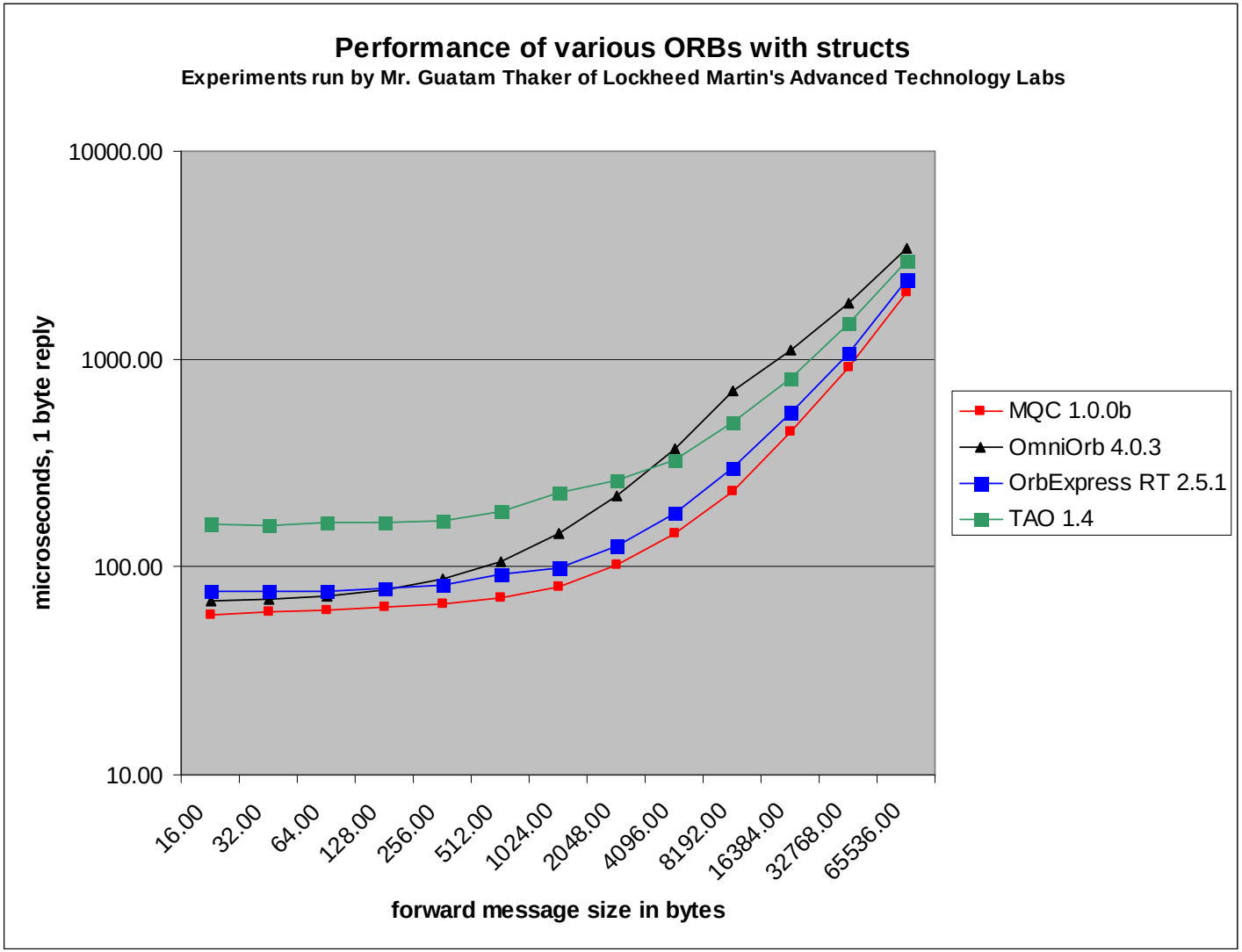
Benchmark #1

- Following is an Excel graph from Guatam's tables:



Benchmark #2

- Following is an Excel graph from Guatam's tables:



RealTime Issues

- Deterministic networking & middleware
  - ?? ARINC 664 and similar deterministic ethernet
  - Issues: Implementation tradeoffs, application API
- Real-Time Scheduling
  - Issues: devise set of scheduling mechanisms and threading models with good coverage (semantics, memory usage, performance)
- Composition of above
  - What do graphs of above mechanisms mean end-to-end?
  - Deterministic networking across routers (including firewalls)
  - Implementation tradeoffs and interactions with fault tolerance, security

Other Items

- Prof. Anneliese Andrews (Separate slides)
- Discussion, Q&A (no slides)

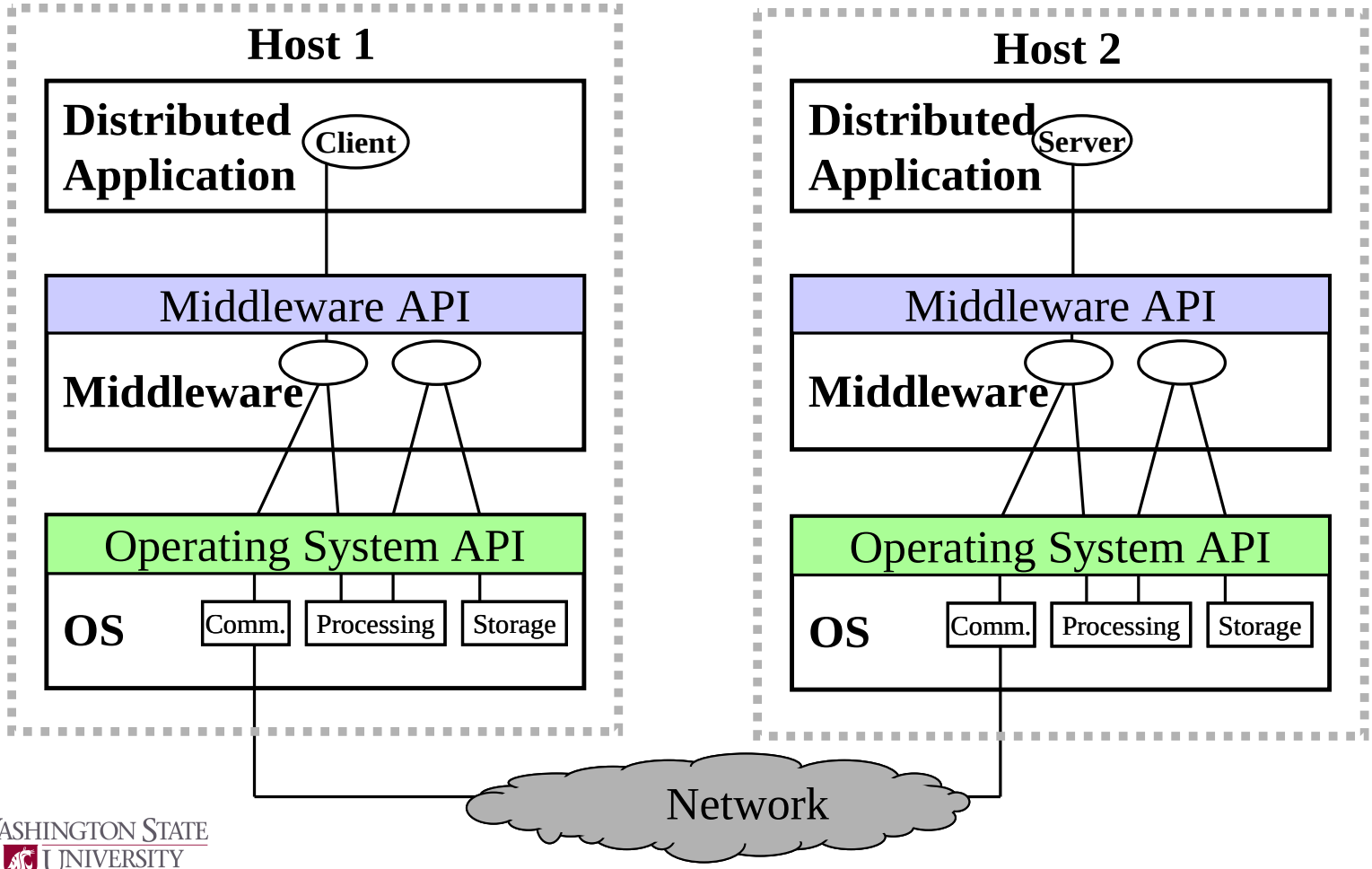
Backup Slides

Outline

- **Introduction**
  - Middleware
  - Embedded systems
  - Quality of Service
- Related Work
- MicroQoSCORBA
  - Middleware Architectural Design Taxonomy
  - Fine-grained Configurable Middleware Framework
  - Embedded System Security
  - Security Subsystem Design & Implementation
  - Experimental Evaluation
- Conclusions

Middleware

"A layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system"



Middleware: Heterogeneity & Transparency

- Middleware's programming building blocks mask heterogeneity
  - Makes programmer's life much easier!!
- Kinds of heterogeneity masked by middleware
  - Heterogeneity in network technology always masked
  - Heterogeneity in host CPU always masked
  - Heterogeneity in operating system (or family thereof) usually masked
  - Heterogeneity in programming language usually masked
  - Heterogeneity in vendor implementations sometimes masked
- Middleware can provide transparency with respect to distribution:
  - Location transparency
  - Replication transparency
  - Mobility transparency
  - Concurrency transparency
  - Failure transparency
- **Masking heterogeneity and providing transparency makes programming distributed systems much easier to do!**

Existing Middleware Frameworks

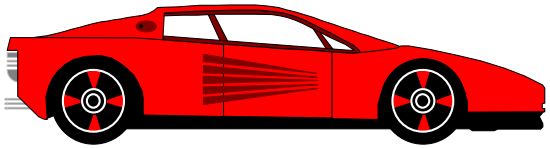
- Support is lacking for
  - Small memory footprint
  - Generality to a wide range of hardware devices
  - Power awareness
  - Multi-property QoS (esp. non-RT properties)
  - Fine-grained configurability
  - Software Engineering & Analysis tools

Embedded Systems Market

- 11 Billion CPUs per year

- System size varies
  - Aircraft, PDAs, Home appliances

- Application volume varies
  - Radios, TVs, Satellites



50 Million Cars/Year

- **Application constraints vary**
  - Business applications
  - Sensor Networks
    - Single-purpose sensors → High-end signal processing
    - Environmental monitoring, battlefield networks , …

Quality of Service

- Real-World applications have real-world tradeoffs!
- QoS Properties
  - Security
    - Multiple strengths/Algorithms
  - Fault Tolerance
    - Quantity and types of faults tolerated
  - Real-time Behavior
    - Scheduling algorithms, Network performance

- Resource Issues
  - Memory footprint
  - Power awareness

Outline

- Introduction
- **Related Work**
- MicroQoSCORBA
  - Middleware Architectural Design Taxonomy
  - Fine-grained Configurable Middleware Framework
  - Embedded System Security
  - Security Subsystem Design & Implementation
  - Experimental Evaluation
- Conclusions

Related Work—CORBA

- MinimumCORBA
  - Removes support for dynamic interfaces, etc
  - Reduces the memory footprint of an ORB (~in half)
- Real-time CORBA
  - Provides tools to better predict time delays
  - Enables hard real-time CORBA applications
- Fault Tolerant CORBA
- CORBA Security Service
- Smart Transducers Interface
- Stand-alone specifications—they do not compose

Related Work, cont.

- Java Remote Method Invocation (RMI)
  - Lacks cross-language support, configurability, QoS mechanisms
- Small Footprint
  - legORB — UIUC, ~ 6 Kb Client IIOP engine
  - e*ORB — Vertel, ~ 35 Kb Client ORB
  - ORBlite — HP Labs: evolvability flexibility, subsetting
  - Stripped-down, *Point solutions*
- VEST
  - Application specific operating systems
  - Lightweight components
  - Strong analysis toolkit
  - Aspects for RT performance & dependability

Related Work, cont.

- MMLite
  - object oriented DCOMcomponents (coarse grained)
  - primary QoS property is RT via scheduling comp.
- QoS and/or Reflection
  - QuO — QoS contracts, adaptive distributed applications
  - MULTE — wide range of latency & bandwidth
  - Open-ORB Python — reflection, component based
  - dynamicTAO — dynamic adaptation, replaceable concurrency, scheduling, & security

Outline

- Introduction
- Related Work
- **MicroQoSCORBA**
  - **Middleware Architectural Design Taxonomy**
  - Fine-grained Configurable Middleware Framework
  - Embedded System Security
  - Security Subsystem Design & Implementation
  - Experimental Evaluation
- Conclusions

MicroQoSCORBA Objectives

- Support wide ranges of (deeply) embedded H/W
  - Resources vary widely (memory, power, etc.)
  - Home appliances, Sensor networks
- Tailor middleware to both application and hardware constraints, with fine granularity
- Develop a *multi*-property QoS enabled MW framework
- Maintain CORBA interoperability
  - Develop an IDL based framework that interoperates with other ORBs, rather than just another IIOP engine

Lifecycle Time Epochs

Lifecycle Epoch	Constraint Bound	Representative Examples
Design	HW Heterogeneity	Symmetric, asymmetric
	HW Choice	X86, TINI, ColdFire
	Communications HW	Ethernet, Serial, Infrared
	Processing Capability	50 Mhz, 1 Ghz, 8bit, 32bit
	System size	small, medium, large (e.g., transducers to jets)
	Power Usage	line, battery, and/or parasitic power
IDL Compilation	Communication Style	Passive, Pro-active, Push, Pull
	Stub/Proxy Generation	Inline vs. library usage
	Message Lengths	Fixed, variable length messages
	Parameter Marshalling	Fixed Formats
Application Compilation	Space/Time Optimizations	Loop unrolling, code migration, function and proxy inlining
	Library Usage	Static vs. dynamic library linkage
System / App. Startup	Device Initialization	Serial port baud rate, handshaking
	Network Startup	Bootp, dhcp
	Major QoS adaptation	Select between QoS modules
Run Time	Minor QoS adaptation	Adjust QoS parameters

Middleware ArchitecturalTaxonomy

Embedded Hardware	Roles			SW I/O	IDL Subsetting
	Control Flow	Data Flow	Interaction Style		
System Comp. • Homogenous • Asymmetric **HW I/O Support** • Serial, 1-Wire, Parallel, Digital, Ethernet, IrDA, Bluetooth, GSM, GPRS **Resources** • Memory • Power **Processing Capabilities** • 8-bit, 16-bit, …	**Connection Setup** • Initiates Setup • Receive Setup Requests **Service Location** • Hardwired Logic • Config. File • Name Service • Other	**Data Direction** • Bits In • Bits Out • Bits In/Out **Parallelism** • 1 Message in Transit • N Messages in Transit	**Sync** (Send/Recv) **Async** (One-Way Msgs) **Msg Push** **Msg Pull** **Passive** **Pro-Active**	**Data Representation** • CORBA CDR • MQC CDR • … **Protocols** • TCP/IP • UDP • PPP • 1-wire **Direct** (i.e., IIOP) **Indirect** (i.e., IIOP Gateway)	**Message Types** **Parameter Types** **Data Types** **Exceptions** **Message Payload**

Outline

- Introduction
- Related Work
- MicroQoSCORBA
  - Middleware Architectural Design Taxonomy
  - **Fine-grained Configurable Middleware Framework**
  - Embedded System Security
  - Security Subsystem Design & Implementation
  - Experimental Evaluation
- Conclusions

Architectural Considerations

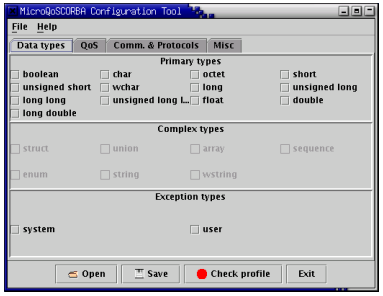
- Fine-grained composability
- Baseline Functionality
  - CORBA IDL support
  - Autogenerated Code (e.g., stubs & skeletons)
- QoS Functionality
  - Security (to be presented later)
  - Fault Tolerance
  - Timeliness
- Development Tools

Fault Tolerance Mechanisms

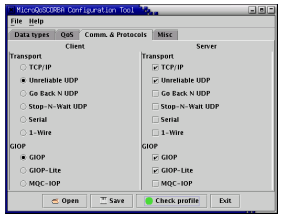
Redundancy	Reliability	Ordering
• Temporal – Multiple transmits • Spatial – Multiple Channels – Replicated Servers • Value – Checksums, CRC	• Group Communication – Best Effort – Reliable – Atomic – Uniform • Failure Detection	• Sender FIFO • Causal – Logical Timestamping • Total – Sequencer / Token based

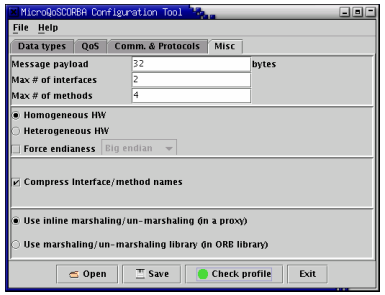
Development Tools

- Not all developers are created equal
- **Goal:** Make it easy for the casual programmer
  - Domain expert, but QoS novice
  - Lifecycle support personnel
  - Temporary/contract employees



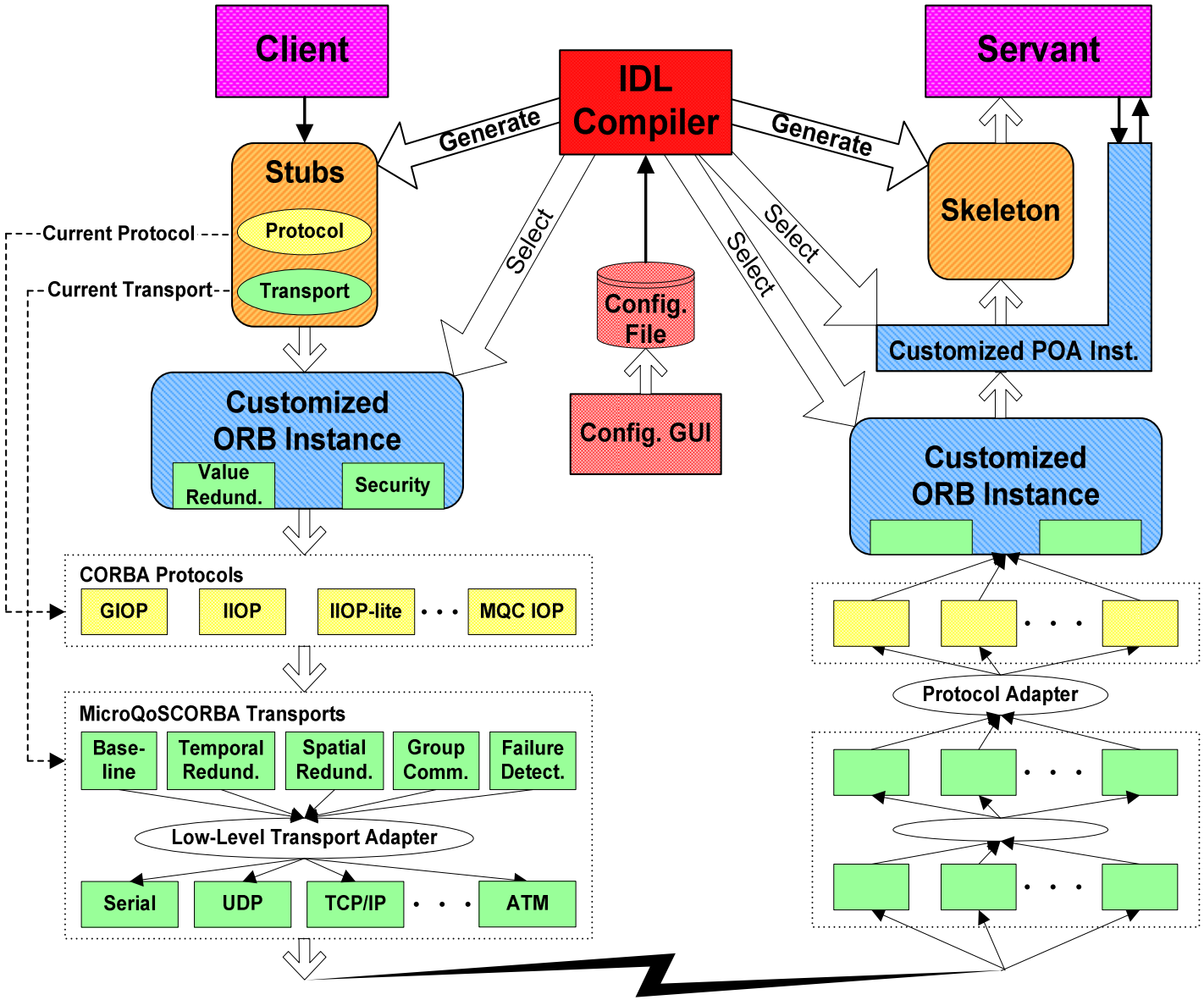
- Tools choose compatible components based upon
  - QoS requirements
  - Resource configuration





- Application and Hardware specific configuration file for the IDL Compiler
  - IDL compiler custom-generates stub/skeleton code

Architecture Overview



Outline

- Introduction
- Related Work
- MicroQoSCORBA
  - Middleware Architectural Design Taxonomy
  - Fine-grained Configurable Middleware Framework
  - **Embedded System Security**
  - Security Subsystem Design & Implementation
  - Experimental Evaluation
- Conclusions

Embedded System Security

- Security must be designed in
- Deep coupling with a physical environment
  - Exposure to the elements, tampering, etc.
- Significant tradeoffs between
  - Security
  - Cost & Resource Usage
  - Generality / Adaptability
- Relatively limited computation power
- Denial of Service attacks are more acute

Security Design Space

Confidentiality	Integrity	Availability	Accountability
Physical —Dedicated Wire —Secure Network Encryption —Symmetric* AES DES Rot13 —Public Key RSA Elliptic Curves	Message Digests* —MD4/5 —SHA Message Authen-tication Codes* —HMAC Error Control/ Correction Codes —CRC32* Digital Signatures —DSA —RSA	Service Continuity *—See Fault Tolerance* Disaster Recovery	Authentication —Physical Tokens —Shared Secrets* —Passwords —Challenge/Resp. Authorization —Access Controls —Data Protection Audit Non-Repudiation

* Currently Implemented

Some aspects of this design space are beyond the scope of MicroQoSCORBA (e.g., Dedicated networks, authentication tokens, PKI infrastructure)

Outline

- Introduction
- Related Work
- MicroQoSCORBA
  - Middleware Architectural Design Taxonomy
  - Fine-grained Configurable Middleware Framework
  - Embedded System Security
  - **Security Subsystem Design & Implementation**
  - Experimental Evaluation
- Conclusions

Configurable Security Subsystem

- Initial Prototype
  - Caesar & AES Ciphers
- Implementing additional mechanisms
  - Reused existing cryptographic mechanisms
    - Cryptix Java Cryptography Extensions (JCE) mechanisms
    - Substantially rewrote the Cryptix JCE class hierarchy
  - Implemented “low-cost” mechanisms
    - XOR cipher
    - Parity & CRC message digests
- Security mechanisms are enabled/disabled via MicroQoSCORBA’s macro mechanisms

MicroQoSCORBA Security Mechanisms

- Supported Ciphers
  - XOR, Caesar, CAST5, DES, 3DES, IDEA, MARS, RC2, RC4, AES, Serpent, SKIPJACK, Square, Twofish
- Support Message Digests
  - Parity, CRC32, MD2, MD4, MD5, RIPEMD, RIPEMD128, RIPEMD160, SHA0, SHA1, SHA256, SHA384, SHA512, Tiger
- Supported Message Authentication Codes
  - HMAC is supported with the above MDs

Outline

- Introduction
- Related Work
- MicroQoSCORBA
  - Middleware Architectural Design Taxonomy
  - Fine-grained Configurable Middleware Framework
  - Embedded System Security
  - Security Subsystem Design & Implementation
  - **Experimental Evaluation**
- Conclusions

Supported Platforms

- Linux Workstation
  - Pentium 4, 1.5 GHz, 256 MB RAM

- Systronix SaJe
  - 100 MHz aJile aJ-100 CPU, 1 MB RAM
  - Native Java execution

- TINI
  - 40 MHz DS80C390 CPU, 512 KB RAM
  - (~equiv. 100 MHz 8051)
  - Emulated JVM (slow)

- PDAs (soon)

Automated Tools

- Necessitated by MicroQoSCORBA's fine-grained configurability of both functional and QoS properties
  - (i.e., literally *hundreds* of configurations were evaluated)
- Complex Makefile targets
  - Update configurations, Execute IDL compiler, Build configurations, Archive builds for later execution
- Expect scripts
  - Automate the performance testing of multiple configurations
  - Scripts developed for the Linux, SaJe, and TINI platforms
- Analysis Routines
  - Common file formats, Expect logs

Example Application

- Timing Example
  - Very simple
  - Note: MicroQoSCORBA has not been completely optimized for memory usage or run time performance

- CORBA IDL

```
module timing {
  interface foo {
    long bar(in long arg1);
  };
};
```

Memory & File Size Comparisons

- MicroQoSCORBA and JacORB are Java based
- TAO is a C++ ORB

Sizes (on Linux)	MicroQoSCORBA		JacORB		TAO	
	Client	Server	Client	Server	Client	Server
Application	4,222 B	2,476 B	6,591 B	6,363 B	33,422 B	66,635 B
Java Mem.	153,560 B	160,648 B	222,968 B	243,120 B	n/a	n/a
Linux RSS	9.95 MB	9.62 MB	13.31 MB	11.46 MB	4.68 MB	5.96 MB

Baseline Application Size & Memory Usage

MicroQoSCORBA Java Class Size (bytes)	Linux Client	Linux Server	SaJe Client[1]	SaJe Server[1]	TINI Client	TINI Server
Baseline	63,607	61,062	259,077	254,246	23,867	20,764
Baseline w/Temp.Red.[2]	59,478	58,617	258,437	252,819	22,506	19,928
Value Redundancy	68,687	66,278	262,506	257,726	25,726	22,675
AES Cipher	85,182	82,634	270,606	265,762	35,156	32,062

Multi-Property QoS Application Size (bytes)

[1] Note: SaJe sizes include runtime

[2] Note: impl. restricts to fixed length msgs so simpler code

Timing Example Latency

Latency	Linux (Filtered/Raw)	SaJe (Filtered/Raw)	TINI (Filtered/Raw)
MicroQoSCORBA	**0.170 / 0.171**	**4.162 / 4.425**	**248.6 / 256.8**
JacORB	**0.329 / 0.604**	n/a	n/a
TAO	**0.330 / 0.332**	n/a	n/a

Baseline (non-QoS) Timing Latency (ms)



QoS Property	Linux			SaJe			TINI		
	No TR	TR2	TR4	No TR	TR2	TR4	No TR	TR2	TR4
Baseline	**0.170**	**0.184**	**0.182**	**4.162**	**4.163**	**4.225**	**248.5**	**241.8**	**340.9**
Value Red.	**0.178**	**0.194**	**0.195**	**4.635**	**5.184**	**5.309**	**415.2**	**465.8**	**687.6**
AES-128	**0.207**	**0.229**	**0.222**	**5.945**	**7.941**	**7.137**	**858.2**	**1,050.2**	**1,554.5**
AES-192	**0.202**	**0.222**	**0.222**	**6.173**	**8.246**	**9.859**	**951.1**	**1,180.9**	**1,751.5**
AES-256	**0.219**	**0.227**	**0.229**	**6.399**	**8.547**	**10.301**	**1,047.2**	**1,311.7**	**1,945.2**

Multi-Property QoS Timing Latencies (ms)

Results from best of three runs; TR≡Temporal Redundancy

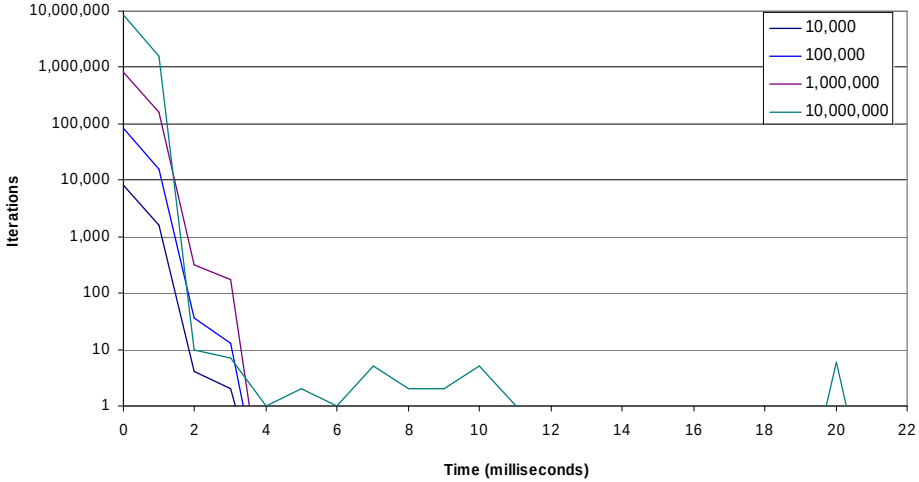
Security Performance



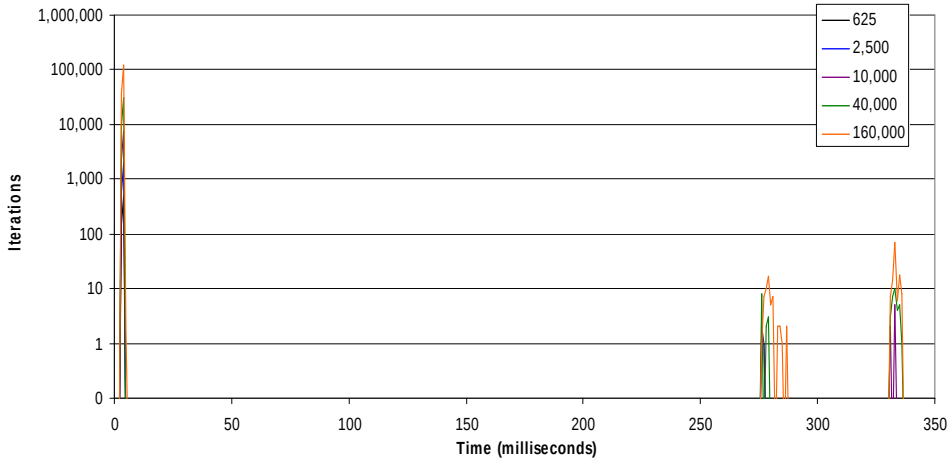
	Security Mechanism	Linux			SaJe			TINI		
		56	512	1024	56	512	1024	56	512	1024
	Baseline	**0.161**	**0.247**	**0.351**	**3.77**	**5.88**	**8.29**	**134**	**129**	**141**
Integrity	**XOR**-8	**0.167**	**0.264**	**0.378**	**4.02**	**8.42**	**13.31**	**198**	**918**	**1,706**
Integrity	**AES**-128	**0.194**	**0.406**	**0.647**	**5.26**	**23.04**	**41.93**	**647**	**6,460**	**12,600**
Integrity	**AES**-256	**0.199**	**0.449**	**0.717**	**5.64**	**28.06**	**51.75**	**805**	**8,545**	**16,690**
Integrity	**DES**-56	**0.190**	**0.538**	**0.914**	**6.15**	**37.84**	**71.53**	**1,052**	**12,916**	**25,500**
Integrity	**3DES**-168	**0.229**	**0.997**	**1.796**	**10.13**	**95.28**	**185.64**	**2,764**	**37,720**	**74,602**
Confidentiality	**Parity**	**0.165**	**0.307**	**0.467**	**4.08**	**8.38**	**13.11**	**233**	**995**	**1,843**
Confidentiality	**MD5**	**0.184**	**0.318**	**0.471**	**6.20**	**17.35**	**29.36**	**919**	**3,870**	**7,010**
Confidentiality	**SHA1**	**0.202**	**0.360**	**0.558**	**6.70**	**19.95**	**34.22**	**1,202**	**5,440**	**9,953**
Confidentiality	**SHA2-512**	**0.315**	**0.878**	**1.466**	**15.30**	**57.32**	**99.48**	**4,168**	**18,713**	**33,186**
Combo	**XOR & Parity**	**0.170**	**0.247**	**0.351**	**4.33**	**10.92**	**18.13**	**299**	**1,787**	**3,414**
Combo	**AES**-128 & **SHA1**	**0.235**	**0.538**	**0.867**	**8.99**	**37.63**	**68.29**	**2,002**	**11,959**	**22,592**
Combo	**AES**-256 & **SHA2**	**0.375**	**1.095**	**1.859**	**19.83**	**82.14**	**145.56**	**5,892**	**28,350**	**51,131**

Security Mechanism Latencies for 56/512/1024 byte messages (ms)

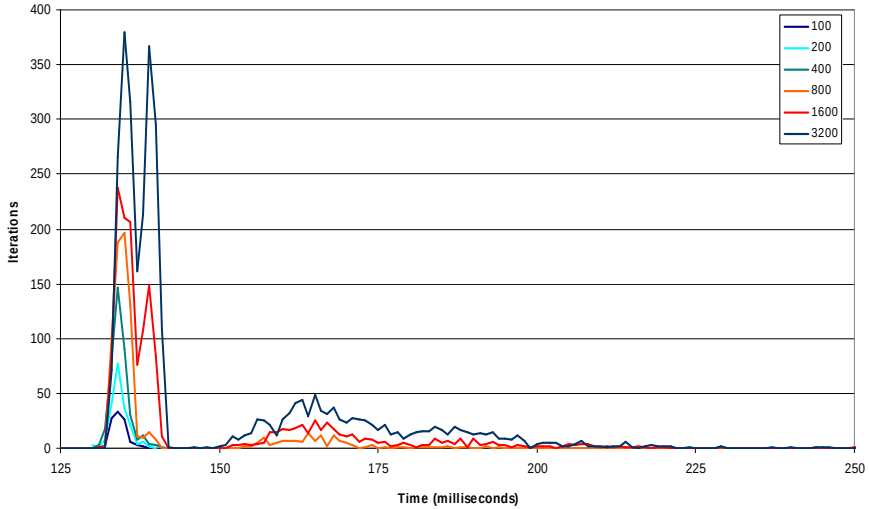
Performance Impacts



Linux ms resolution



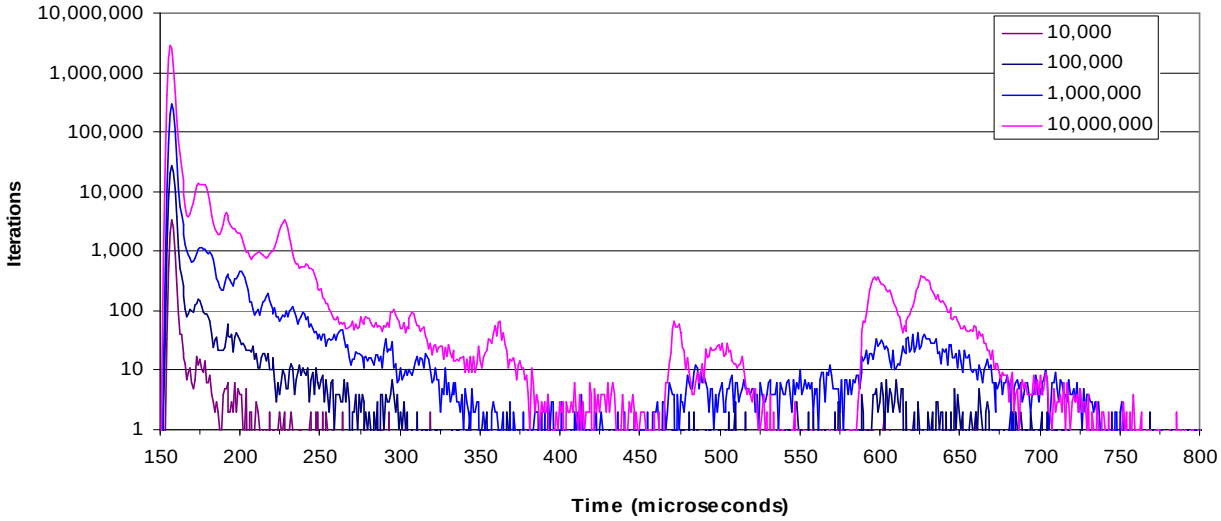
SaJe ms resolution



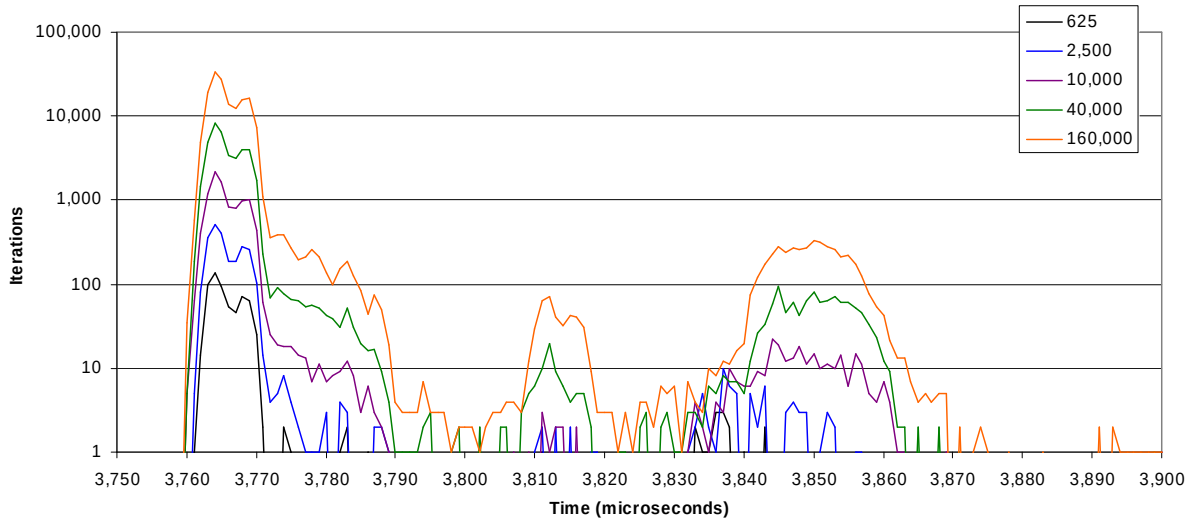
TINI ms resolution

- Java garbage collection and system/network performance impacts best-case performance

Performance Impacts (cont.)



Linux µs resolution



SaJe µs resolution

- On Linux and SaJe the experiments were repeated with µs timer.
- The ms and µs results moti-vate the need for "event filtering".

Outline

- Introduction
- Related Work
- MicroQoSCORBA
  - Middleware Architectural Design Taxonomy
  - Fine-grained Configurable Middleware Framework
  - Embedded System Security
  - Security Subsystem Design & Implementation
  - Experimental Evaluation
- **Conclusions**

Overview of Ongoing/Future Work

- C++ version in progress, late October-ish
- Temporal profiling toolkit
- Additional protocol support (SMTP, IPv6)
- Wireless compensation layer
- IDS mechanisms for embedded middleware
- CASE tools for *(in collab. with Prof. Andrews)*
  - Generation of instrumentation & validation code
  - Aspect oriented QoS+resource constraint management

Dual Use:

MicroQoSCORBA as embedded middleware *and* MicroQoSCORBA for multi-property QoS investigations

Conclusions

- MicroQoSCORBA the only framework that
  - Is a "bottom-up" rethinking from the device level of what should be configurable, and in what ways
  - Is tailorable for a given application to the wide range of
    - Device constraints, ***and***
    - Application-dictated constraints
    - ***with*** a fine granularity of configuration constraints
  - Supports both "functional" and QoS properties
    - Security and fault tolerance as well as real-time performance are all key QoS properties

- … one more important conclusion …