PRISMTECH
Productivity Tools & Middleware

Java For Real-Time Enterprise Systems

Delivering the Benefits of Java to the world of Real-Time distributed object computing

Simon McQueen
CORBA Technical Lead
July 2006

The Case for Java in Enterprise Real-Time Systems

- Changes in scale complexity and scope of Real-Time systems are forcing the Real-time community to re-think the way in which it develops software
- Elements of Real-Time programming are now finding their way into applications that had never previously required Real-Time programming expertise
- It is increasingly difficult to view Real-Time developers and business application developers as members of two separate communities
- Ways have to be found to build bridges between the two communities to facilitate the smooth flow of information from the smallest embedded device to the largest enterprise information systems and vice versa
- These trends make it necessary for the Real-Time community to adopt Real-Time platforms that provide higher level abstractions and advanced tools for functional and temporal programming

Benefits of Real-Time CORBA

- CORBA is well established as a technology for integrating diverse systems
- Used extensively for mission and business critical applications in many industry sectors such as defense, telecommunications and manufacturing
- With the publication of the Real-Time CORBA Specification, the OMG extended the benefits of CORBA to the Real-Time domain
- The standard addresses the issues of end-to-end predictability across CORBA systems and provides a solution in terms of priority control, synchronization and resource control
- Until very recently developers have been forced to implement their applications in programming languages such as C, C++ and Ada that can support the temporal constraints of Real-Time systems

What is OpenFusion RTOrb Java Edition ?

- OpenFusion RTOrb Java Edition is PrismTech's Real-Time CORBA compliant ORB for the Java platform
- OpenFusion RTOrb Java Edition is the first COTS Java ORB that can be used in **"hard"** Real-Time systems
- Made possible through the recent emergence of RTJVM implementations based on the Real-Time Specification for Java (RTSJ)
  - e.g Sun's Java Real-Time System v1.0 (formally known as the Mackinac JVM) & IBM's J9 JVM
- OpenFusion RTOrb Java Edition can provide a unified solution for different needs and uses, supporting both hard Real-Time systems and non Real-Time Enterprise applications in a single ORB

Objectives

- Prove practicality of Real-Time CORBA in a Java environment
- Build a single Enterprise ORB implementation that can in both Real-Time and non Real-Time systems
  - Support for SUN's Real-Time System for Java (formally Mackinac) JVM running on Solaris 10 and IBM's J9 on Linux
  - Use in non RT environment with JSE JVMs
- Providing support for CORBA 3.0 and Real-Time CORBA v1.2 standards
  - Including full enterprise features set, including:
    - *CORBA Messaging – QoS, AMI, Portable Interceptors, ValueTypes, Pluggable transports*
- Performance - high throughput, low latency – as good as JacORB or better
- 100% interoperability with TAO & JacORB
- Real-time acceptance criterion
  - Measured jitter <1mS in test environment

Target Market Characteristics

- Systems requiring advanced software tools for both functional and temporal programming
- Customers who typically have large and complex software integration problems to solve
- Systems where Java is the obvious best long term language choice
- Mission or business critical applications where a hybrid architecture consisting of Real-Time and Enterprise level CORBA functionality is a requirement

Example Application Domains

- **Large Scale Defense Integration**
  - *C4i – wide range of Command Control Computers Communication and Intelligence Systems e.g. Combat Management Systems, Weapon Control Systems*
- **Telecommunications and Networking**
  - *Business management applications*
  - *Operations support systems*
  - *Call control*
  - *Intelligent networking*
  - *STN/Internet convergence*
  - *Integrated network management*
  - *Call and multimedia processing*
  - *Voice Over IP (VOIP) telephony*
- **Aerospace**
  - *Air traffic management*
    - *Saving or displaying management data*
    - *Command interpretation of inputs from a user interface*
- **Manufacturing**
  - *Industrial controllers*
  - *Automotive controllers*
  - *Industrial robotics*

So Why Can't Java Be used in Real-Time Systems ?

- Use of Java in Real-Time systems is limited by its inability to predictably control the temporal execution of applications due to:
  - Unpredictable latencies introduced by automatic memory management (garbage collectors)
  - Inadequate scheduling control
  - Unpredictable synchronization delays
  - Very coarse timer support
  - No asynchronous event processing
  - No "safe" asynchronous transfer of control

The Real-Time Specification for Java

- The Real-Time Specification for Java (RTSJ), developed through the Java Community Process by the Real-Time Expert Group
  - Objective to enable the real-time embedded software developer to use the Java programming language where predictable/hard real-time behaviour is a must
- Specified as a set of extensions to the Java Technology Model
  - For example, the RTSJ shall not prevent existing properly written non real-time Java programs from executing on implementations of the RTSJ

Key Extensions to the Java Technology Model

- The RTSJ supports the following important new features which make Java suitable for use in “Hard” Real-Time systems:
  - New Real-Time Threads, Scheduling, and Synchronization
  - New Memory Management Schemes
  - Asynchronous Events Handling & Asynchronous Transfer of Control
  - Time & Timers
  - Direct Access to Physical Memory

ORB Design Challenges

- Architecture must support dual mode of operation, full RTSJ/Real-Time CORBA mode and conventional ORB mode
- Full Enterprise ORB functionality must also be available in RTSJ mode
- In RTSJ mode ORB designed to use No Heap Real-time Threads, Scoped & Immortal Memory throughout the call chain
- Using RTSJ memory models is non trivial and complex
  - Careful decisions have to be made about which parts of the ORB are long lived and run in Immortal memory and which parts of the ORB are more dynamic and can run in Scoped memory
    - NHRT threads typically allocated in immortal memory, with critical sections of code executing in scoped memory
  - Memory assignment rules must be strictly followed so as to avoid potential dangling references
- Exceptions cannot be thrown in scoped memory and caught in a different memory areas
  - Use of specific design patterns required to propagate exception information from ORB to the application
- ***Note: ORB design issues also highlight the same issues faced by the application programmer!***

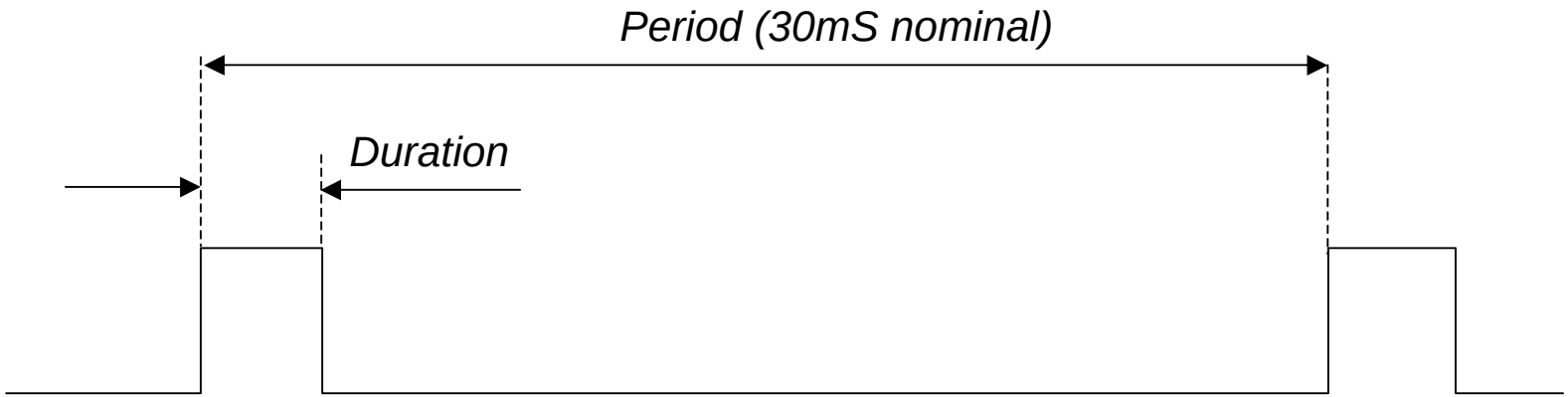
Development Approach

- An experimental approach with three phases:

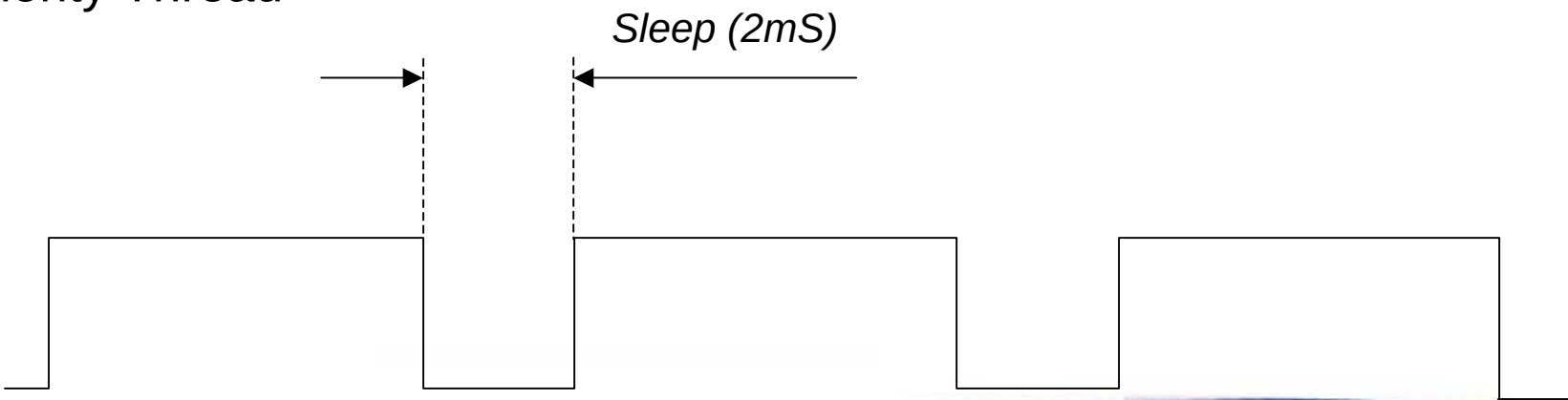
1. Obtain benchmark figures from a C socket to socket example
2. Repeat socket example in Java
   - Isolate contributors to jitter
3. Repeat tests with PrismTech's OpenFusion RTOrb Java Edition
   - Develop tests using RT threads, NHRT Threads, Scoped and Immortal memory

Test Application

High Priority Thread

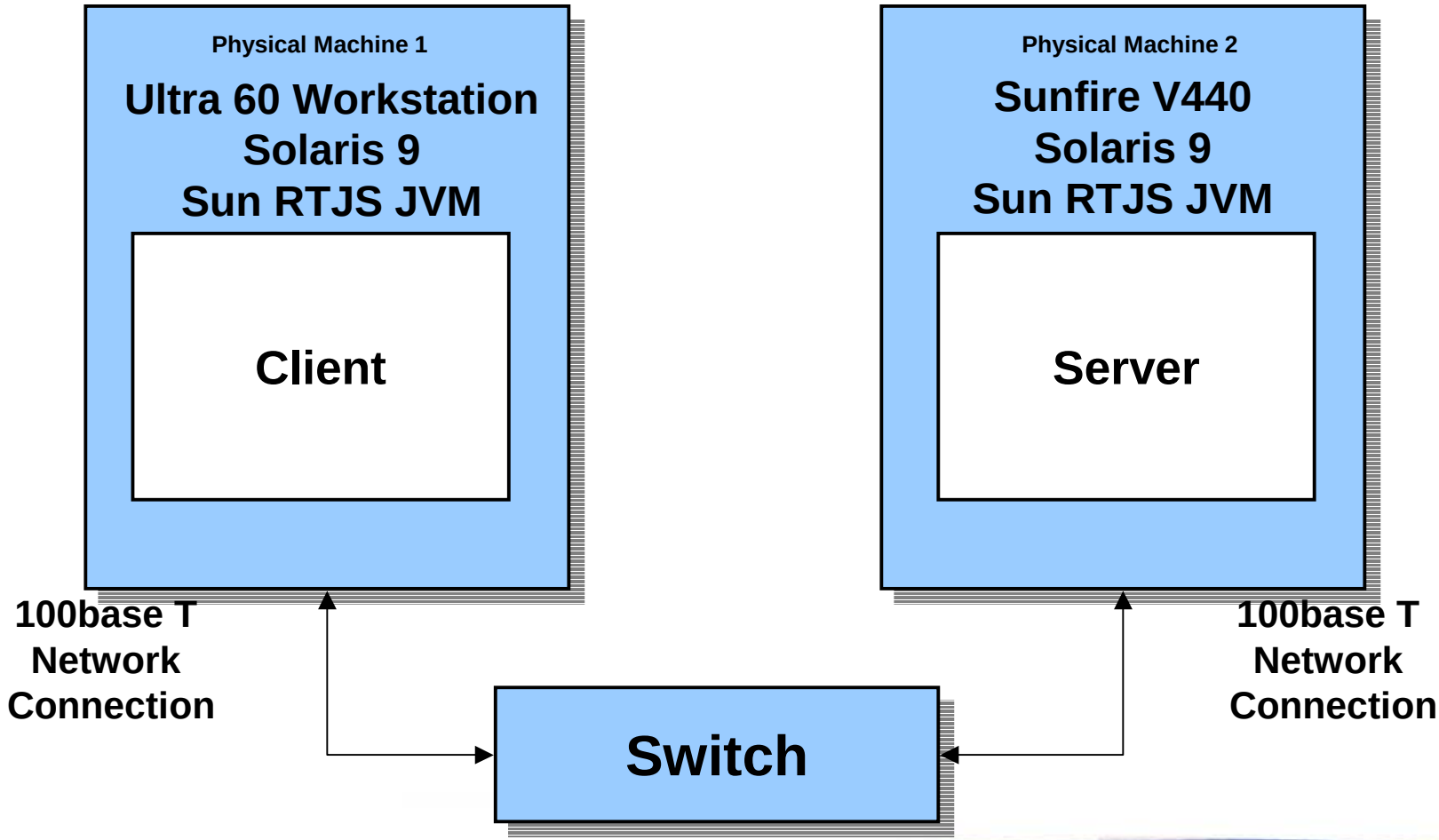


Low Priority Thread

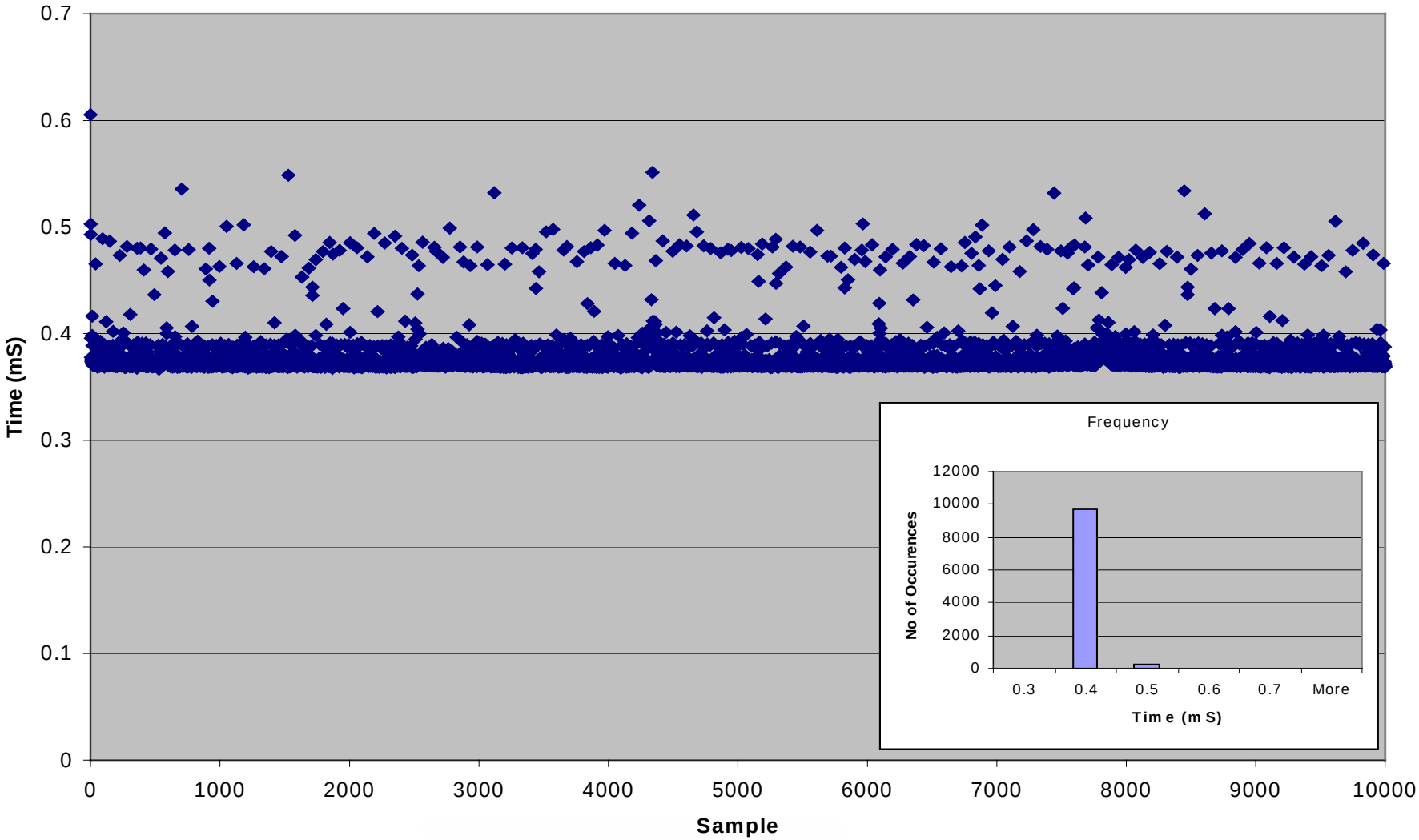


Test Configuration

Networked client and server



C Client Server, 1 High Priority Thread

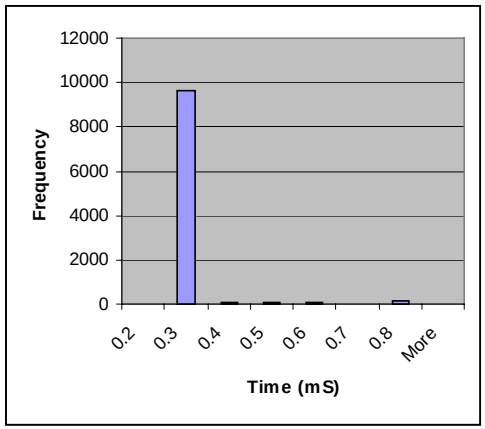


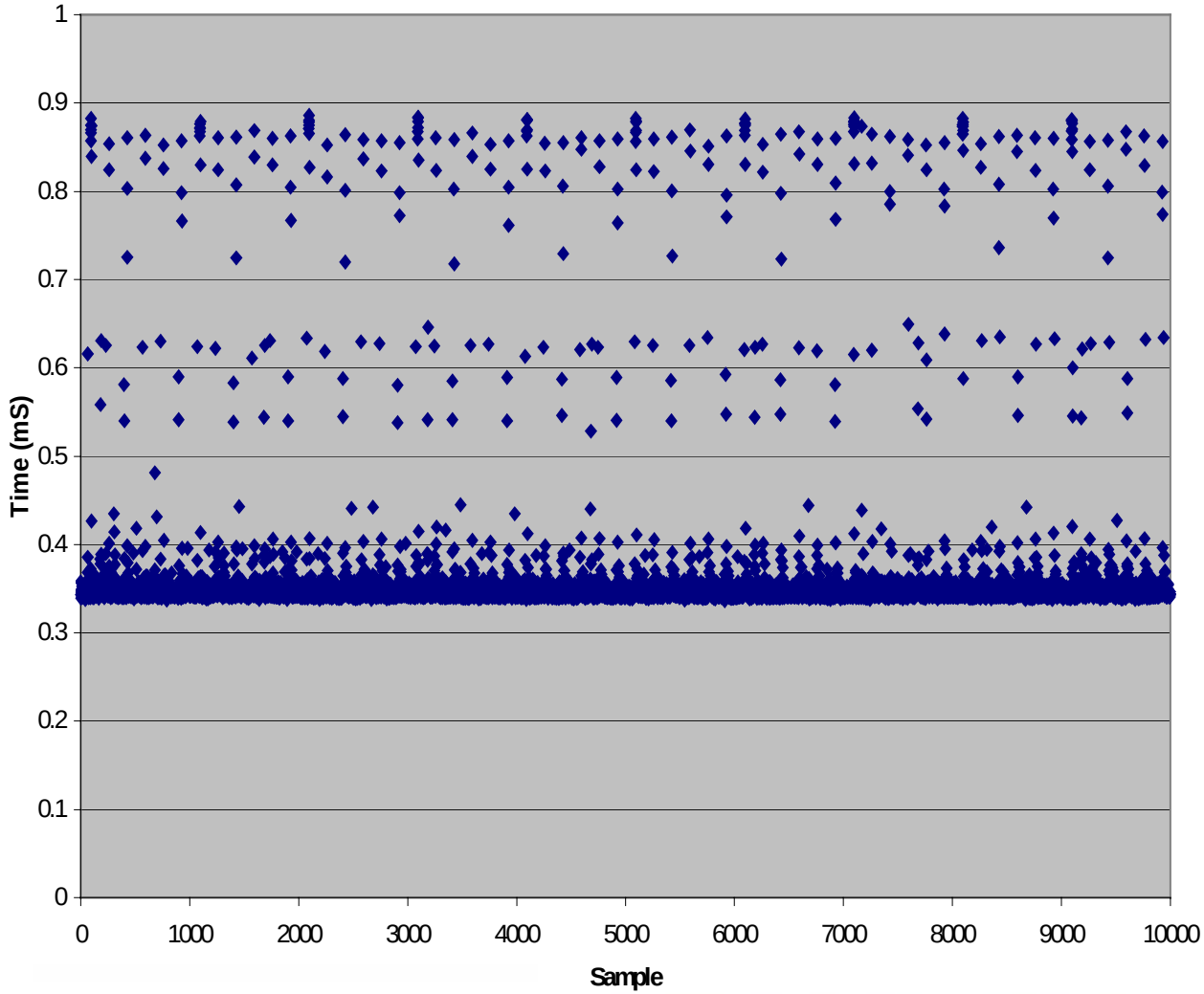
Co-located Java Sockets

Last 10K results of a 20K run

GC logged over run – only minor events recorded

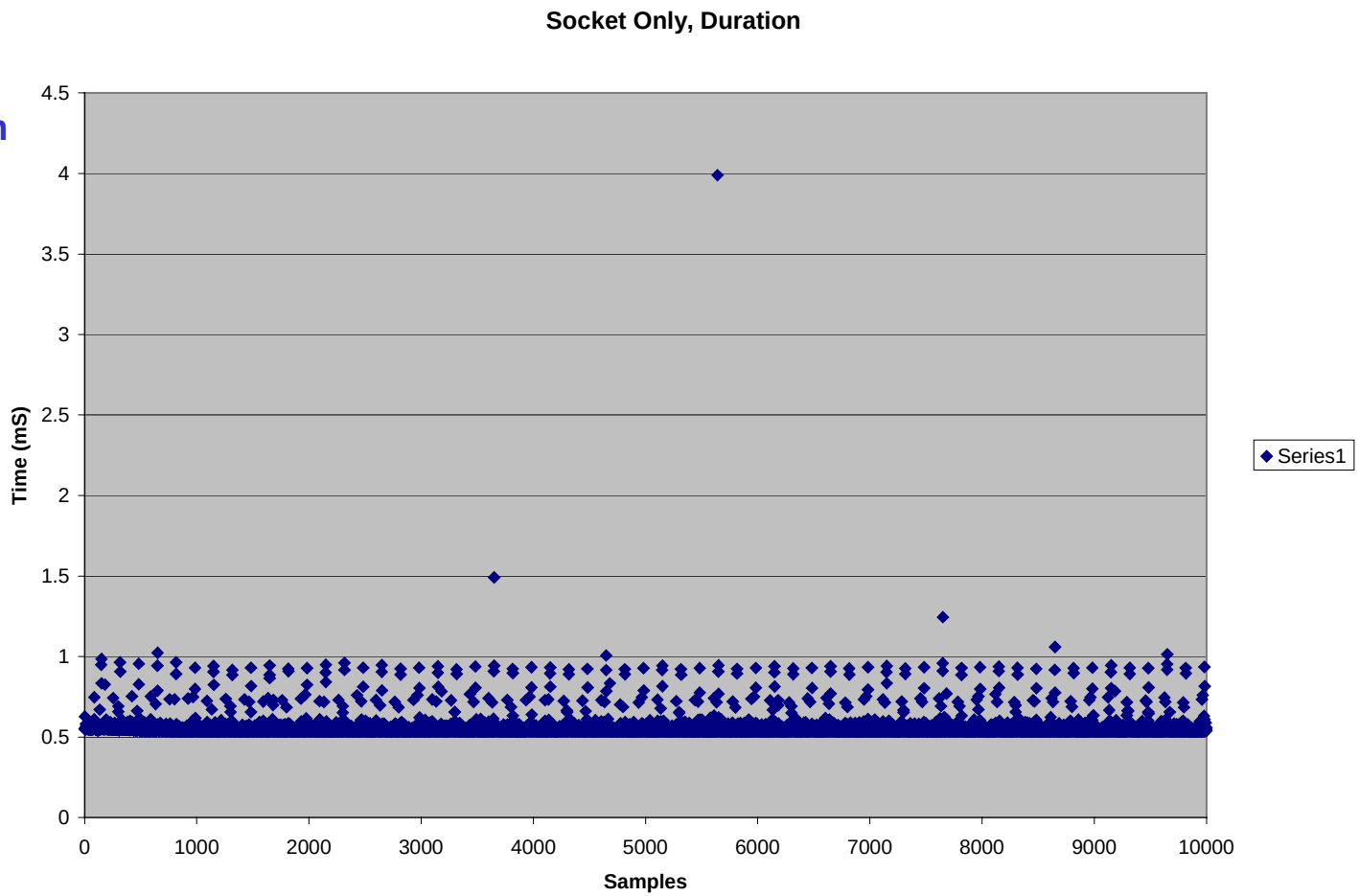
All results within 0.6mS





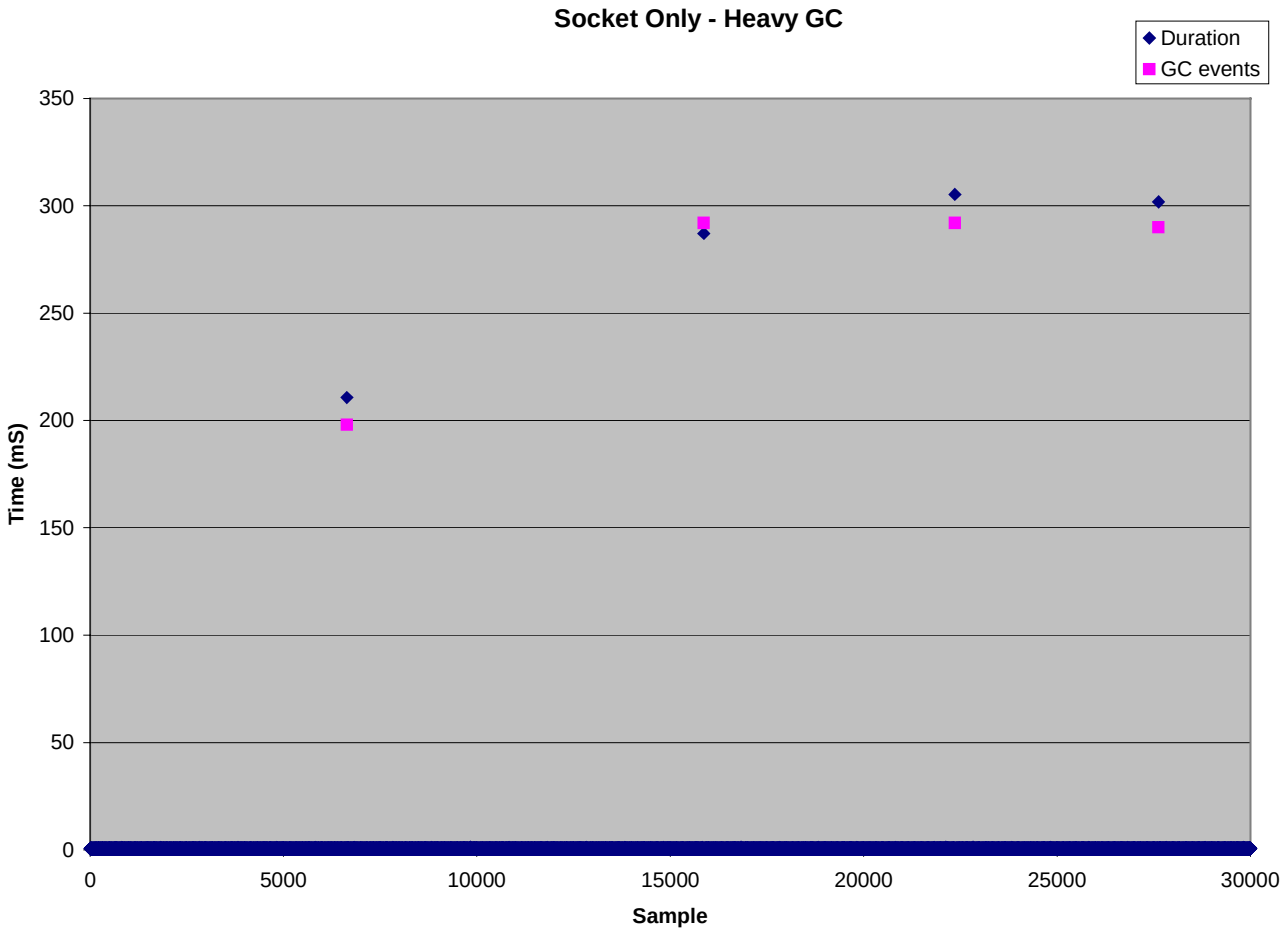
Networked Java Sockets

- Last 10000 samples from a 50000 sample run.
- Heap size 256MB
- No Garbage Collection recorded during entire run
- 95% of results within 0.1mS
- Only 0.03% of results outside 0.5mS
- Indicates what can be achieved

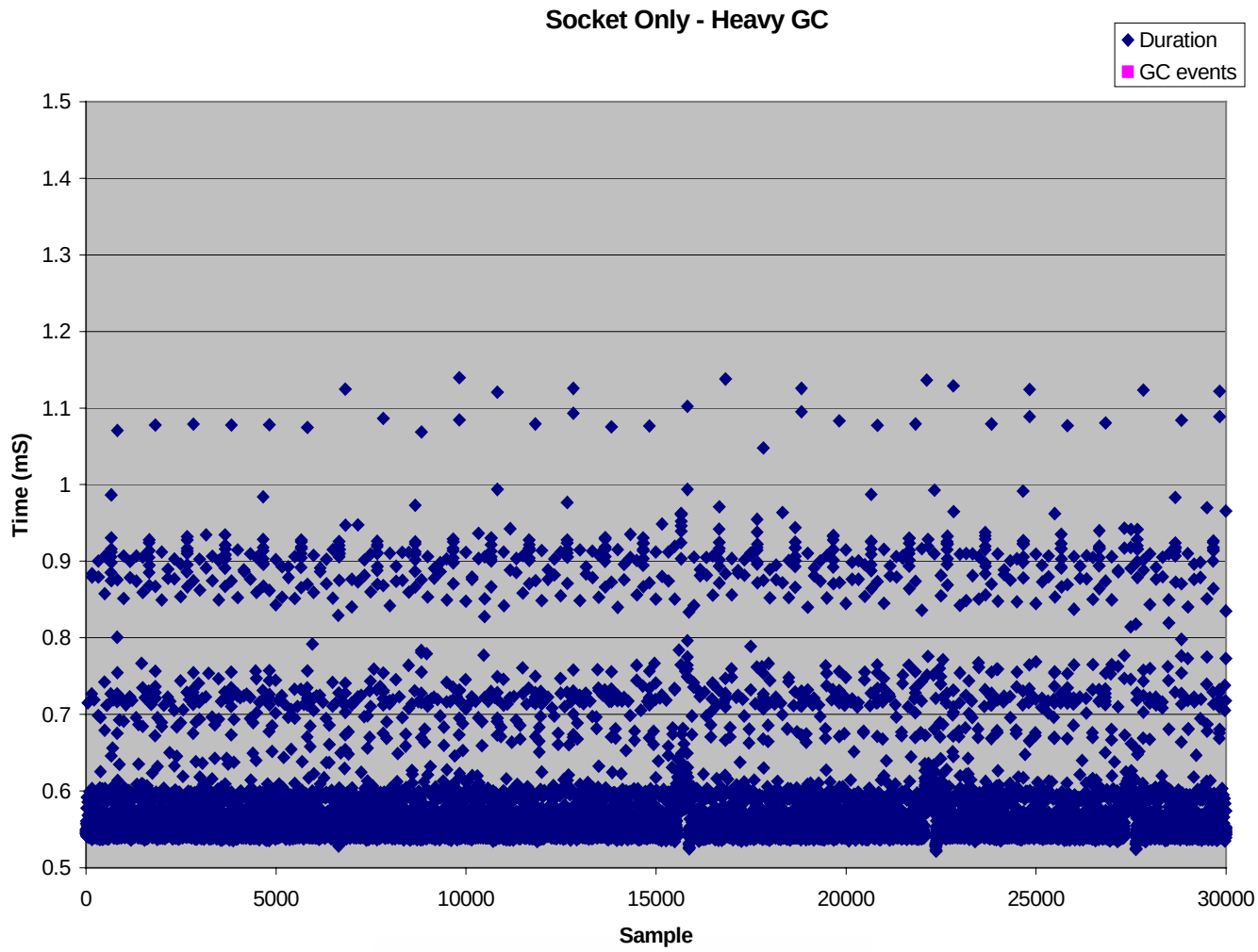


Sockets and Heavy GC

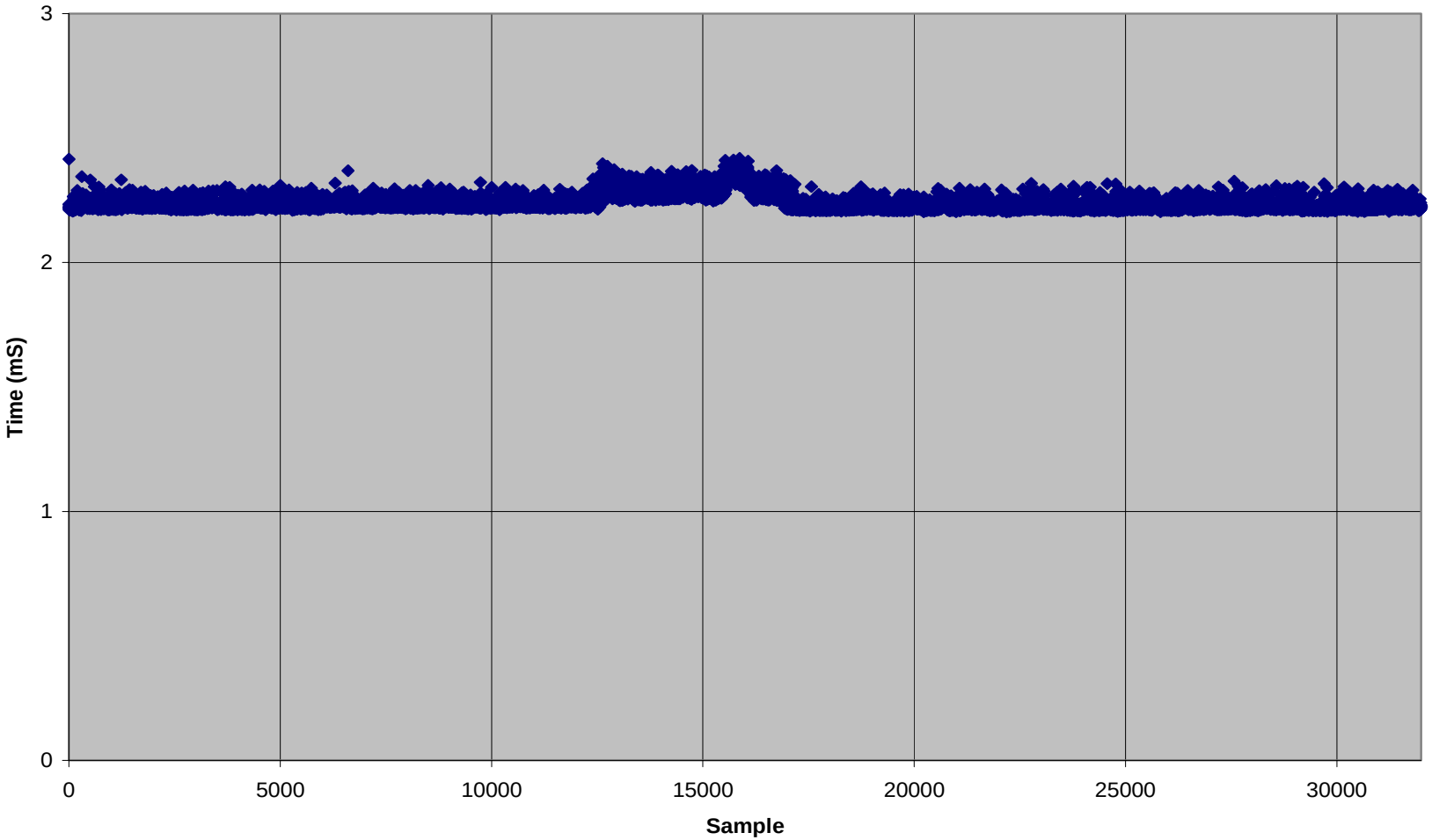
- Last 30K samples of 40K run
- Heap size 4MB
- Dramatic increase in GC effects



Socket only, Heavy GC



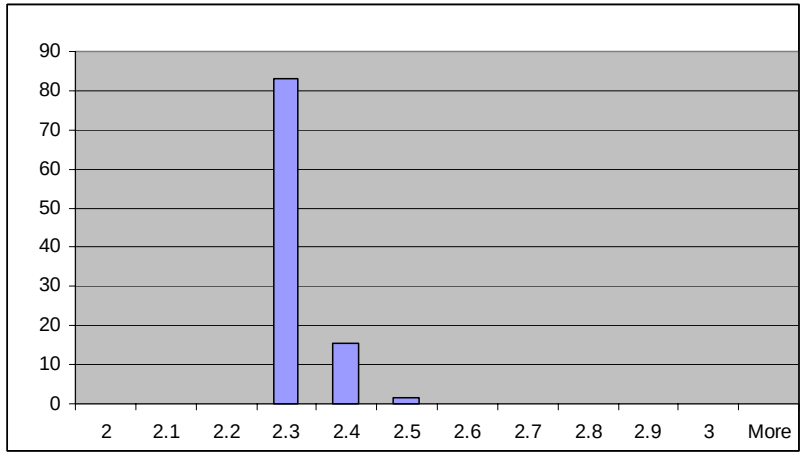
Real-Time Predictability – Low Jitter < 1mS



Results showing the roundtrip CORBA call invocation times for a high priority thread with one or more low priority threads running concurrently

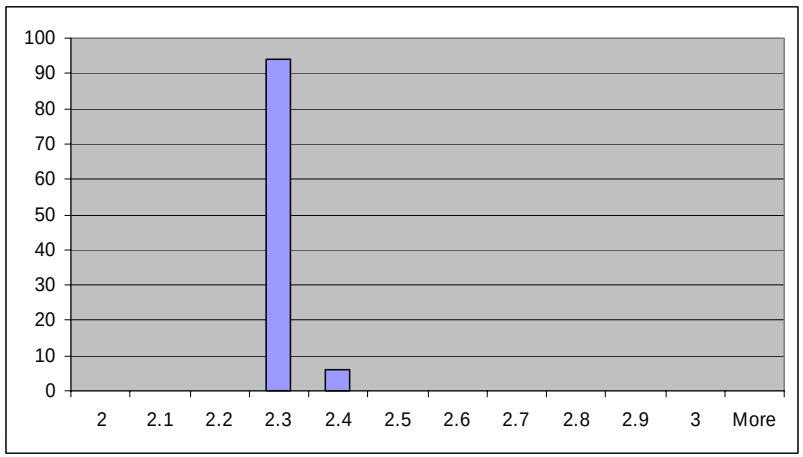
Distribution of Duration

One Hi Priority task only

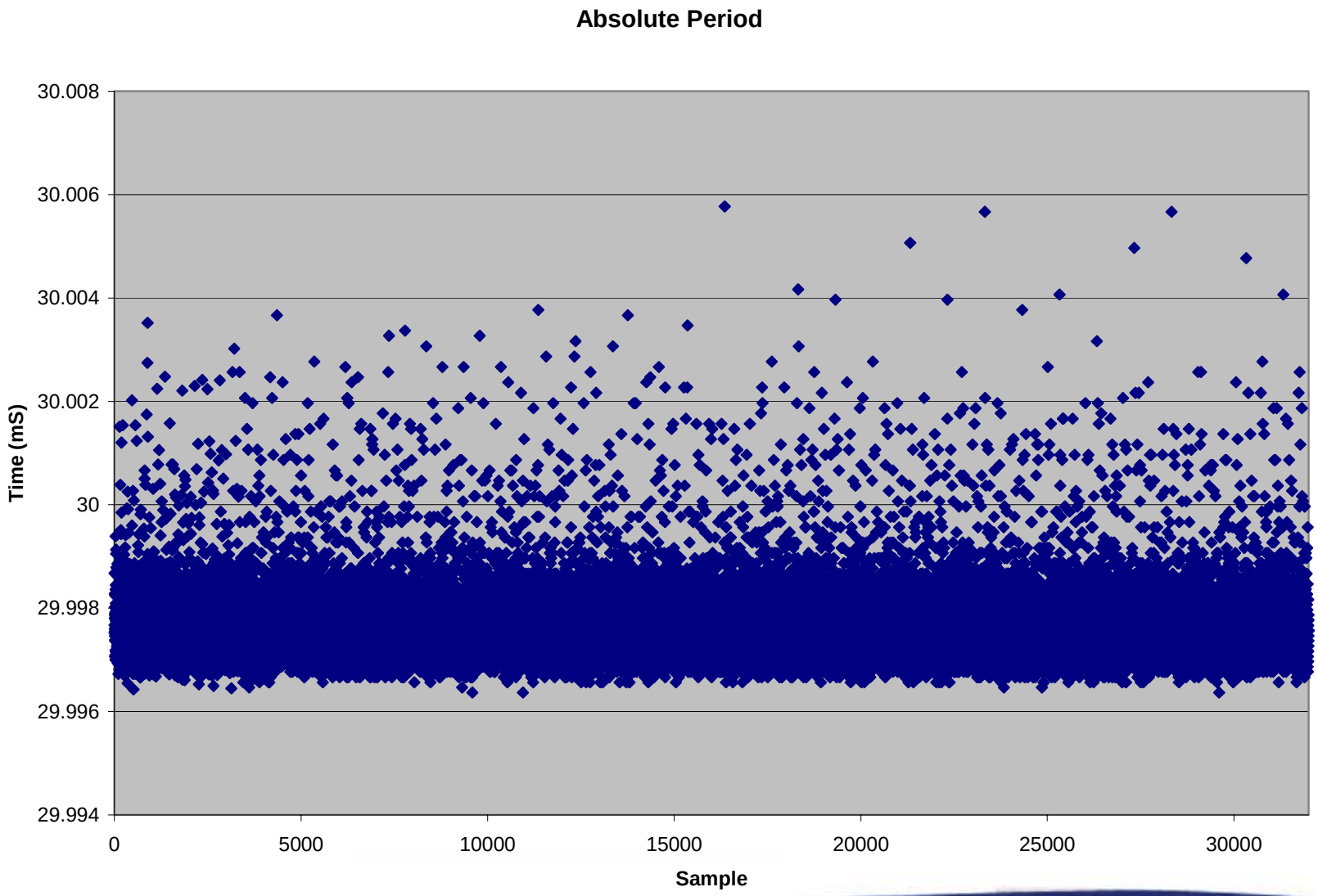


One Hi Priority task

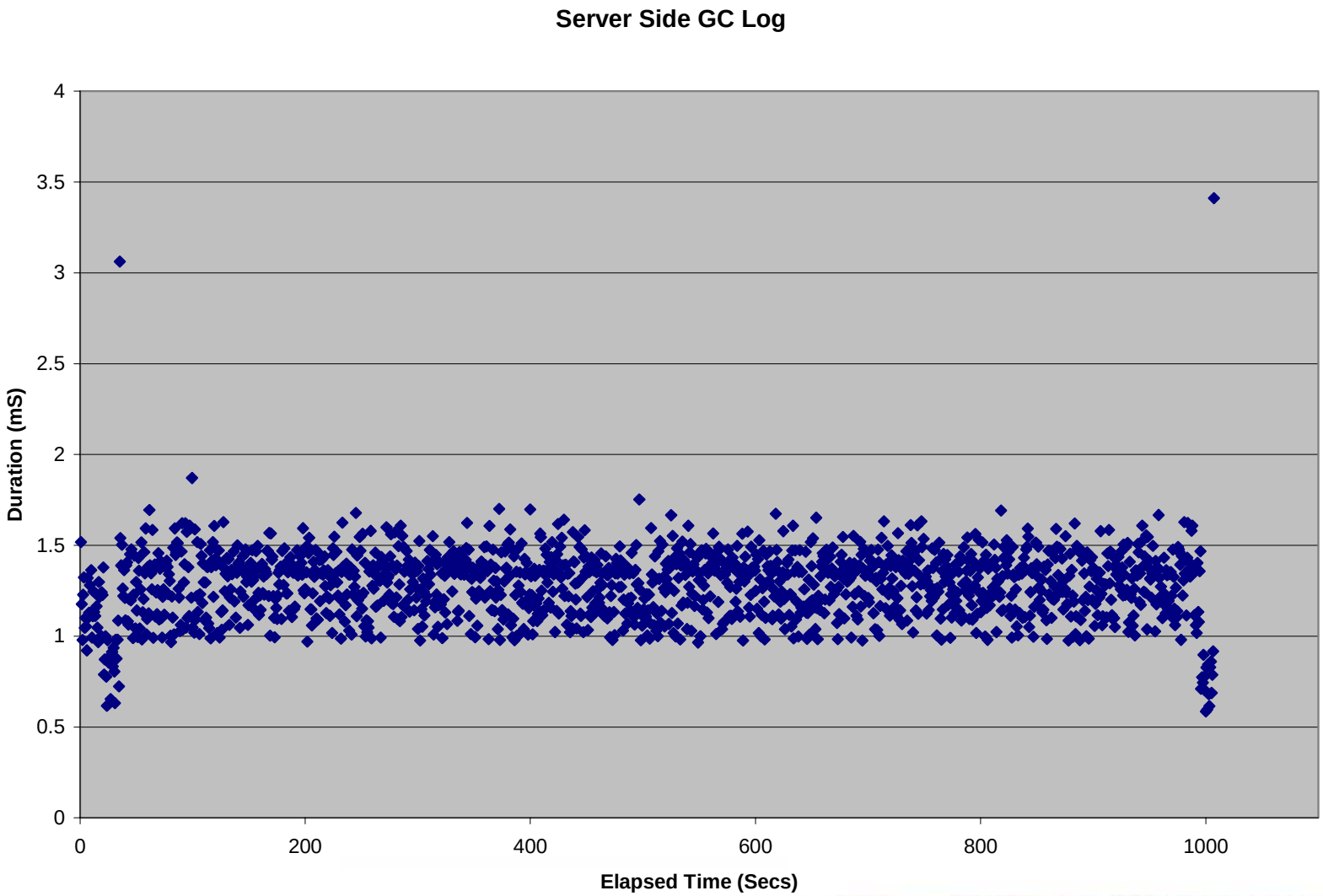
One Low Priority task



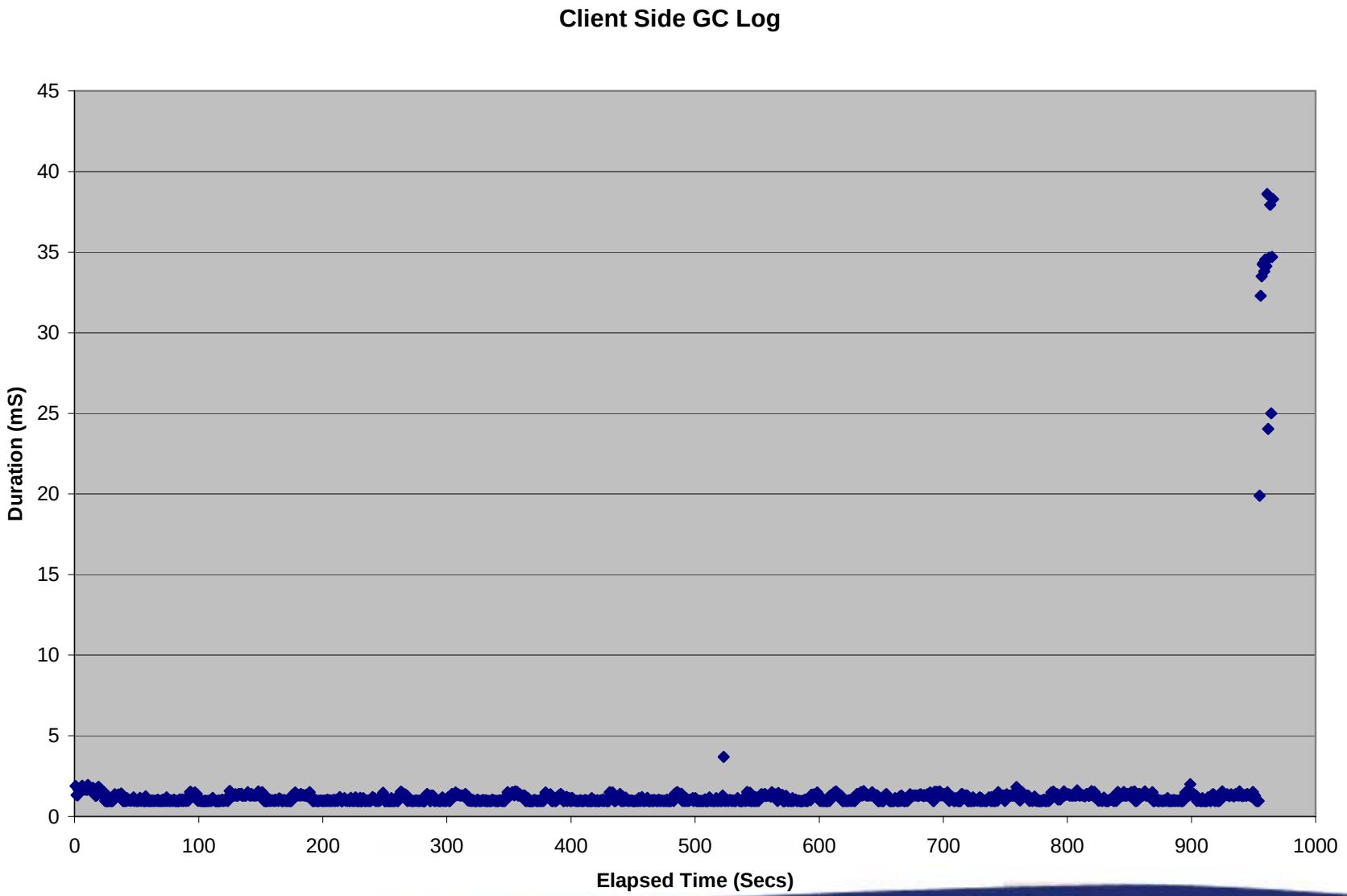
RT ORB, High and Low Priority Tasks



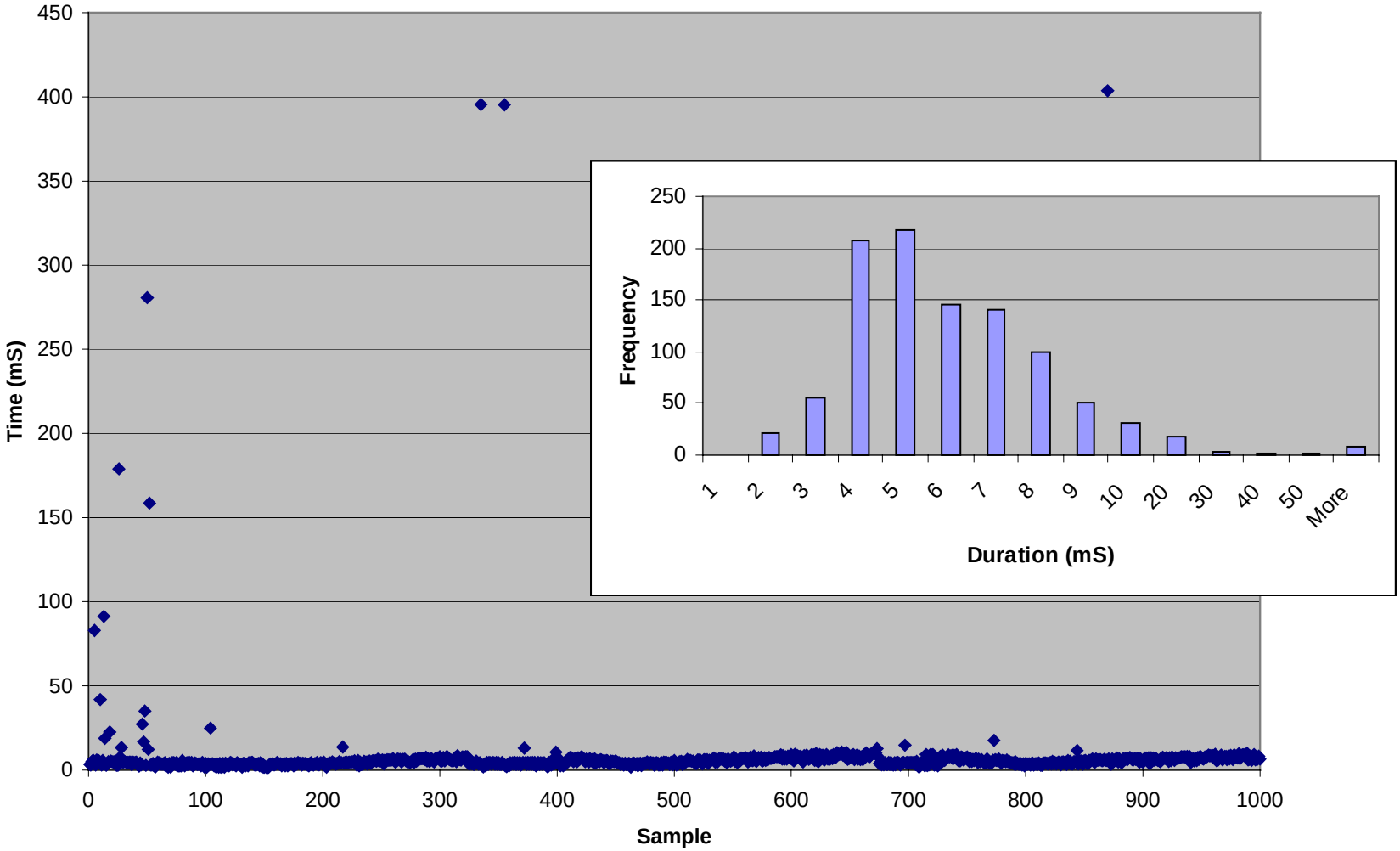
Garbage Collection Log - Server



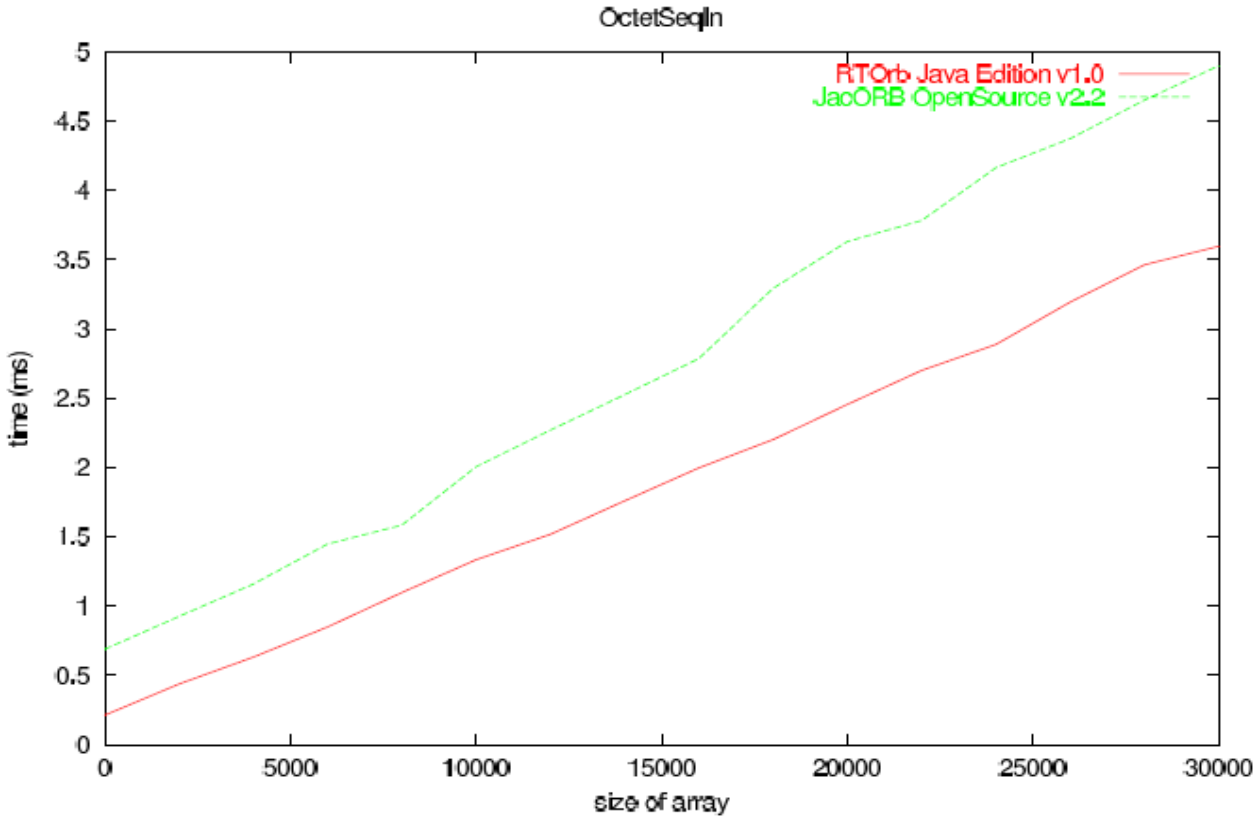
Garbage Collection Logs - Client



Non-Real-Time ORB and JVM

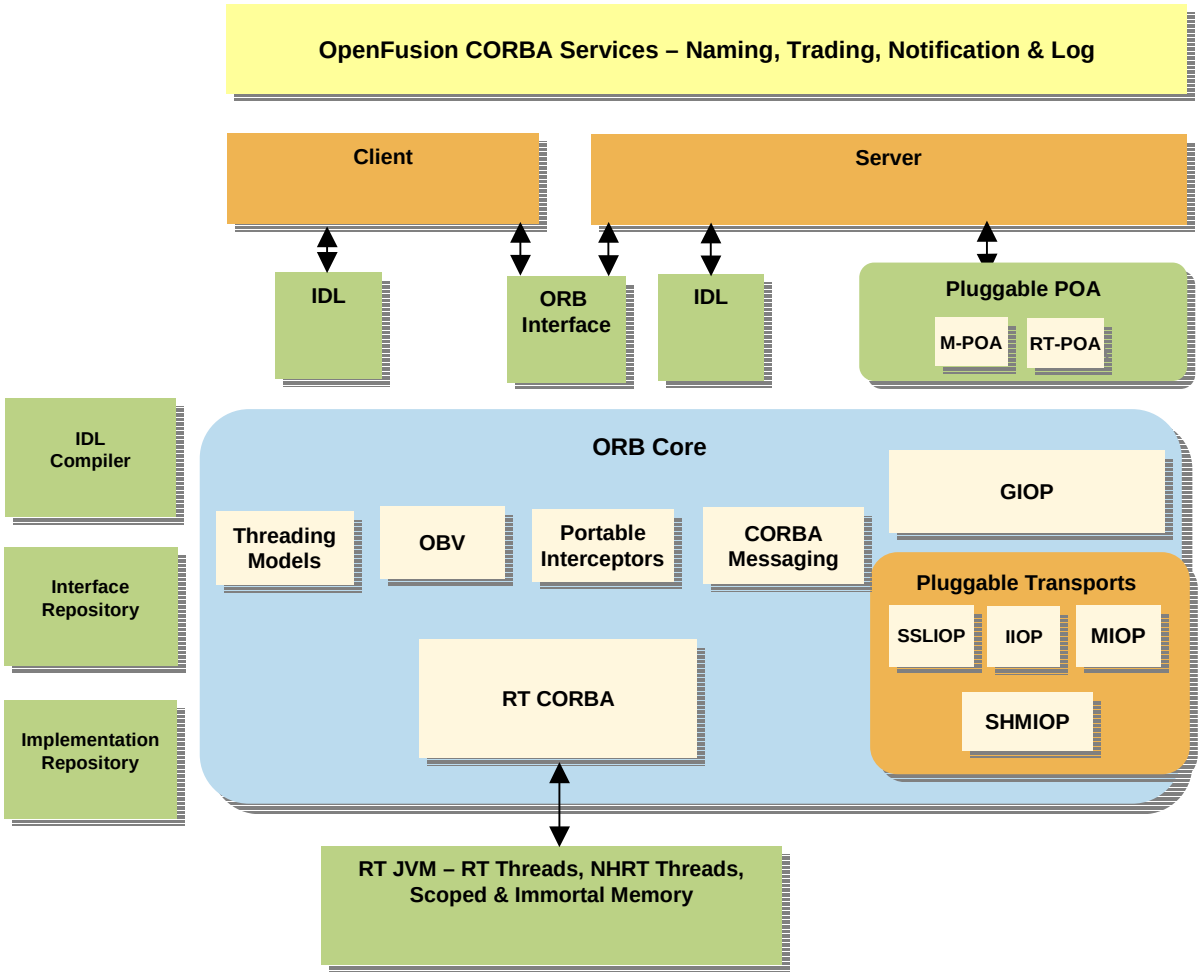


Excellent Performance – Low Latency, High Throughput



Client to server roundtrip invocation times for a series of calls with increasing payload. All measurements carried out on Solaris 10, single host inter-process communication, using JDK 1.4.

OpenFusion RTOrb Java Edition Architecture & Features



- **CORBA 3.0 ORB** . Including features such as:
  - IDL to Java compiler, GIOP 1.3, Full POA, Dynamic ANY, Portable Interceptors, CORBA Messaging (AMI, QoS), Value Types
- Support for OMG's **Real-Time CORBA v1.2** specification. Including features such as:
  - RT ORB, RT POA, RT Priority and Mappings, RT Current, RT Priority Model – Client propagated and Server declared, Priority Transforms, Mutex , Thread Pools, PrivateConnection Policy, Invocation Timeout
- Pluggable transports – support for OMG's **Extensible Transport Framework** (ETF)
- **Common Object Services** - OpenFusion Naming, Trading & Notification Services
- **JVM** - Sun Real-Time Java System v1.0, JDK 1.4, IBM J9
- **Operating System Support** - Solaris 10, Linux, LynxOS

Resulting Benefits

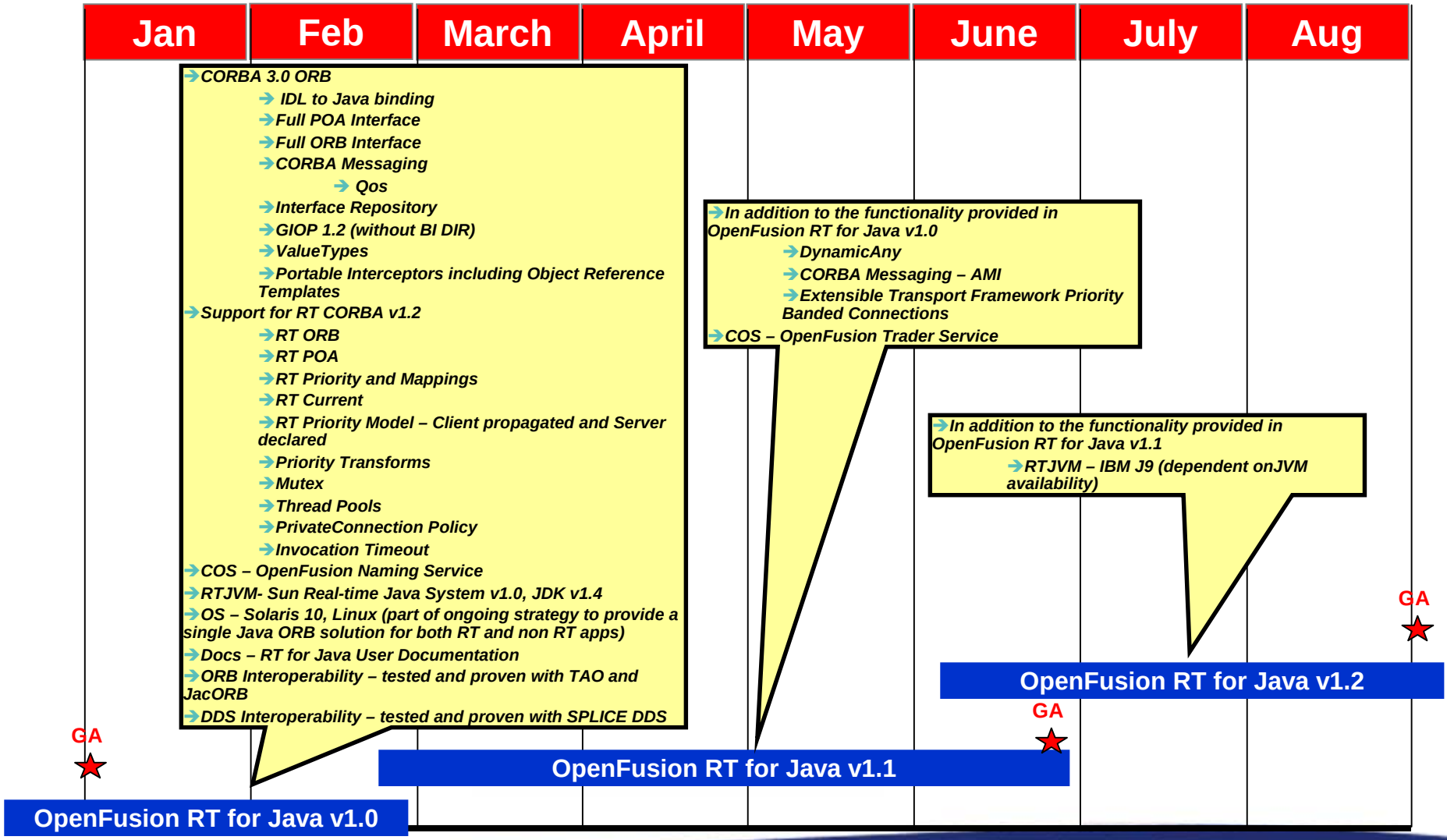
- First commercial CORBA implementation that makes Real-Time Java programming in a distributed environment possible
- Enables Real-Time system developers to leverage the benefits of the Java platform – write once run anywhere portability, ease of use and security
- Compliant with key standards from the OMG and the JCP:
  - CORBA 3 and Real-Time CORBA v1.2 Specifications
  - RTSJ (Real-time Specification for Java v1.0)
- Full enterprise level CORBA functionality
- Can be used as a Real-time ORB or general purpose Enterprise solution or both
  - For systems with a mix of uses (both RT and non-RT) provides a single ORB solution
  - Single ORB solution minimises ORB interoperability issues
  - Developers only have to learn how to use one ORB
- Low jitter (< 1ms)
- High performance (excellent latency and throughput characteristics, as good or better than other Java ORBs)
- Guaranteed interoperability with TAO and JacORB

Future Direction

- ORB implementation that can utilize JVMs that provide support for Real-Time garbage collectors e.g. IBM J9
  - May facilitate Real-Time CORBA in a Java environment but without the need for application developers to use complex RTSJ memory models
  - Will enable the application developers to once again leverage one of the key benefits of the Java Programming Model – namely **“ease of use”**
  - Extensive benchmarking required in order to determine what class of Real-Time system can be supported, “soft” or “hard” or both
  - Additional standardization works required at OMG in order to provide a Java language binding more suitable for Real-Time systems
    - Support for object caching in order to reduce amount of garbage generated in standard stubs and skeletons
    - Support for specific RTSJ features such as scoped memory in the generated binding

Networked Sockets with Real-Time Garbage Collection

OpenFusion RTOrb Java Edition Release Schedule



Summary and Conclusions

- Real-Time CORBA in a Java environment is now a reality
- OpenFusion Real-Time Java ORB can provide a single unified ORB solution for use in both Real-Time (hard & soft) and Non Real-time systems
- Obvious application of this technology for complex applications comprising significant non-real-time logic and both soft and/or hard real-time logic where Java is the preferred long term implementation language
- The impact of GC can be eliminated by using NHRT threads, scoped and immortal memory
- Writing Real-Time CORBA applications in Java can be complex and certainly requires careful use of RTSJ features such as scoped and immortal memory as well as Real-Time/NHRT threads
- Real-Time garbage collectors may offer a good potential compromise for support of certain classes of RT system – helping make developing Real-Time applications in Java less complex

Further Information

For further information about OpenFusion RTOrb Java Edition

- Visit PrismTech's Website at:

  http://www.prismtech.com

- E-mail PrismTech:

  info@prismtech.com