The Use of Web Services Technologies in the U.S. Surface Navy Domain

Fred Weindelmayer
Paul Haynes*
Naval Surface Warfare Center, Dahlgren Division

* UK Ministry of Defence Exchange Scientist

Background

Surface Navy Combat System Characteristics

- The set of human and machine resources that comprise the fighting capability of a warship:
  - Sensor systems, weapon systems, communication systems, navigation systems, and command & control (C2) systems
- Large, distributed, real-time (RT) applications (soft and hard)
- Maximum acceptable latencies typically in the 0(10) – 0(100) millisecond range
- Consequences of not meeting requirements for time-deterministic behavior could be severe for the warfighter and others
- Combat systems must undergo comprehensive, rigorous software safety reviews
- Long system life, often spanning several decades
  - A major contributor to life cycle cost

The Navy's Goal

- Is to build a fleet where combat systems…

…are modular, interoperable, and affordable to upgrade

Open Architecture (OA)

- OA is an enabler for meeting this goal

Naval OA *is an enterprise-wide, multifaceted business and technical strategy for acquiring and maintaining National Security Systems (NSS) as interoperable systems that adopt and exploit open-system design principles and architectures.*

OA CORE PRINCIPLES

- ***Modular design and design disclosure***
- ***Reusable application software***
- ***Interoperable joint warfighting applications and secure information exchange***
- ***Life cycle affordability***
- ***Encouraging competition and collaboration***

Source: OPNAV ltr Ser N6N7/5U916276 dtd 23 Dec 05

Web Services Paradigm

- A set of technologies that standardize interactions among applications, such that they can be made independent of computing platform and language
- Based on open standards, built on Web technologies
- Promotes loose coupling among components
- Enables interoperability and reuse (with careful design)
- Basis for implementing a Service- Oriented Architecture (SOA)
- Well aligned with Open Architecture principles and goals

Web Services Standards

- Core:
  - XML, SOAP, WSDL, HTTP, UDDI
  - Java 2 Enterprise Edition (J2EE) has built-in support for these
- Web Service (WS) extensions (WS-*):
  - WS-Reliable Messaging
  - WS-Security
  - WS-Notification – pub/sub
- Major standards participants:
  - W3C
  - OASIS
  - Sun Java Community Process (JCP)
- Quality of Service (QoS) mechanisms needed to ensure RT requirements are met for combat systems

Note: This list is not meant to be exhaustive.

Comparison of QoS Using Data Distribution Service (DDS) and WS-Notification

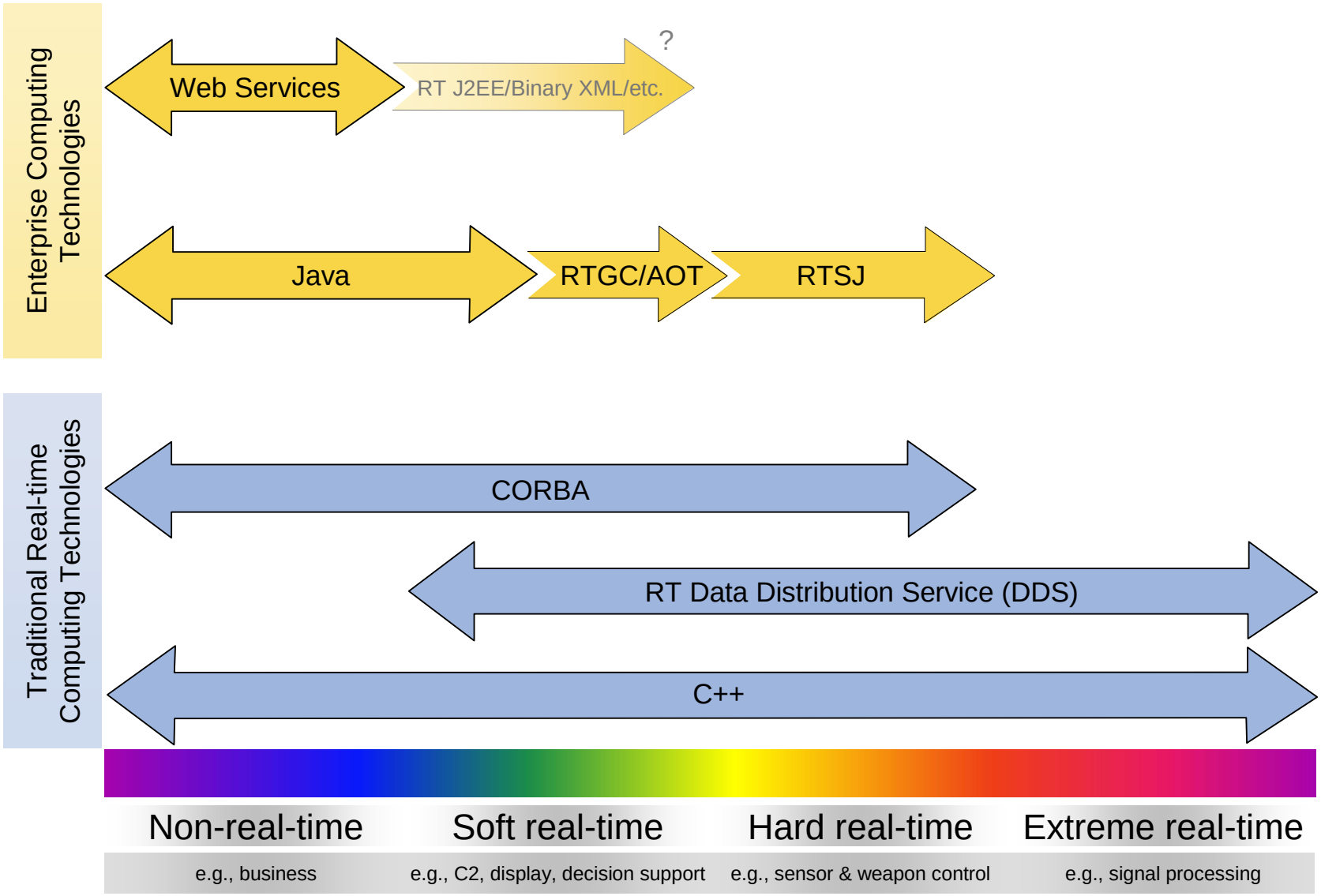
QoS Dimension	DDS	WS-Notification
Location of QoS	QoS built into products	QoS dependent on implementation of other WS-* specs
Discovery	Yes	UDDI
Deadline	Yes	No
Durability	Yes	No
Distributed Process Management	No	WS-Distributed Management
Liveliness	Yes	Possibly, see WS-Resource Framework
Ordered Messages	Yes	WS-Reliable Messaging
Reliability	Yes	WS-Reliable Messaging
Information Assurance / Security	No, but individual products may support	WS-Security
Topics	Yes	WS-Topics

Using Web Services in Combat Systems

- Combat system architectures could benefit from Web Services and their inherent SOA attributes
- The Web Services community is developing some of the QoS standards that we need
- There are opportunities for Web Services technologies to facilitate the integration and interoperability of Navy combat systems with the wider Global Information Grid (GIG)
- <u>The Big Questions:</u>
  - Since Web Services were not designed for RT systems, can the Navy reconcile this paradigm with the demands of RT combat systems and use this technology base to better achieve the goals of open architecture?
  - If so, where will these technologies best be suited in combat systems?

Technology Landscape

Real-Time and Enterprise Technologies & Standards Comparison



Challenges to Adoption of Web Services in the Real-Time Domain

- RT J2EE frameworks
  - Could the recent advances of J2SE products with RT capabilities such as RT Garbage Collection (RTGC), honored thread priorities and Ahead-of-Time (AOT) compilation percolate up to the J2EE domain?
- XML messaging in RT
  - XML provides a good mechanism for describing and giving context to data. However, large, verbose text messages lead to throughput and latency problems in bandwidth-constrained environments
  - There are processing penalties incurred in parsing through XML messages to access data and pass it to the application
- Use of HTTP for transport
  - Built upon TCP – delays from retransmission timeouts, packet reordering, etc.
  - Could other mechanisms (e.g., DDS) be used to efficiently carry SOAP messages to and from J2EE application servers / SOAP engines, facilitating Enterprise computing in the RT domain?

Emerging Real-Time Web Services Technologies

- J2EE application servers with RT capabilities
  - Exploration and development in this technology area is ongoing
  - Built upon recent vendor RT Java frameworks (RTGC, AOT, and/or RTSJ)
- Binary XML
  - Emerging W3C Efficient XML Interchange standard
  - W3C Wireless Application Protocol Binary XML standard
- RT databases (RT DBs)
  - In-memory databases

Real-Time Considerations with Web Services

- Middleware
  - Performance and determinism of RT J2EE application servers
  - Ease of use and composability of WS-* implementations
  - Suitability and efficiency of sending SOAP with middleware technologies
  - JDBC / DB latencies

- XML
  - Latencies and throughput using XML over HTTP
  - XML message sizes
  - XML processing latencies
  - XML compression rates and ratios to binary XML
  - Processing latencies for binary XML
  - Ease of use of binary XML
  - Processing latencies for XML transforms (e.g., into objects)

Note: This list is not meant to be exhaustive, but is a sample set of characteristics that would need to be considered.

Experiment Design

Objectives of the Experiment

- Obtain information on Web Services frameworks and technologies that can be used in Surface Navy combat systems
  - For example, given the plethora of technologies and frameworks, which ones are best suited for the needs of combat systems in the near and long term?
- Assess the present benefits and limitations of Web Services technologies
  - Even though most technologies are lacking RT capabilities, the Navy needs to assess how well current performance characteristics meet combat system requirements
- Gain insight into the usage of Web Services, and determine what technologies and features need further advancement in order to support combat system RT requirements

Scenarios

- Scenario #1
  - 100+ tracks sent per second
  - Client sends tracks to track service
  - Track service processes and stores track data in data store
- Scenario #2
  - 50 track requests per second
  - Client requests track from track service with unique request ID
  - Track service sends the corresponding track data back to client
- Thirty minute data collection periods unless otherwise noted

Experiment Construction

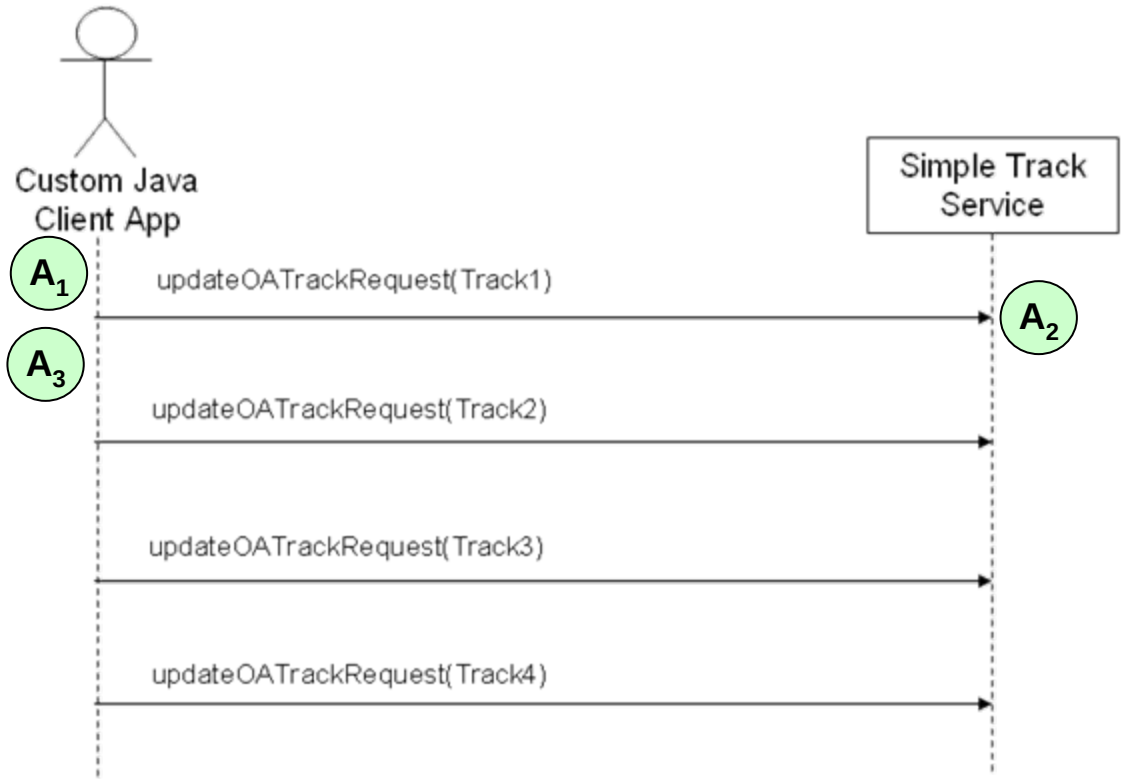
- Simple track service
  - WSDL used to define request/response data types and interfaces for a simple track service
  - Client and service code generated using the WSDL2Java utility
  - Service resides on Apache Axis2 SOAP engine (“out of the box” settings)
  - Client is a standalone Java application
  - Transport: HTTP

- Platform configuration

	Client Platform	Service Platform
Hardware	2 Intel Xeon 3.06 GHz CPUs, 1GB RAM	Sun Fire X4100, 2 AMD Opteron Model 280 (2.4Ghz/1MB) Dual Core Processors, 8GB RAM
OS	RHEL 4.0	Solaris 10
JVM	J2SE 1.5	J2SE 1.5
Web Services Framework	Custom Java Application	Apache Axis2 v1.2

Experiment Design & Instrumentation

Scenario #1: Client sends track updates to track service



Instrumentation

A_1: Client Send Request Time

A_2: Service Receive Request Time

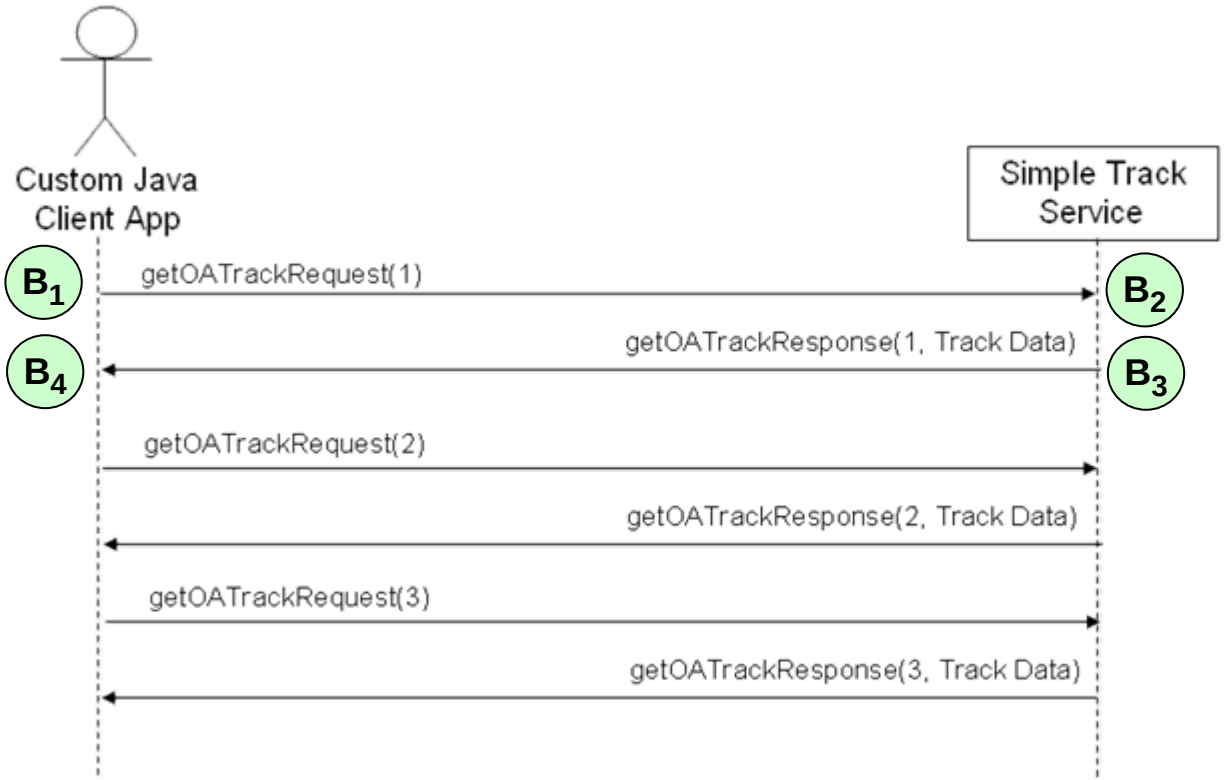
A_3: Client Callback Complete Time

Client Callback Latency = $A_3 - A_1$

Client-to-service Latency = $A_2 - A_1$

Experiment Design & Instrumentation (cont.)

Scenario #2: Client request tracks from track service



Instrumentation

B_1: Client Send Request Time

B_2: Service Receive Request Time

B_3: Service Send Response Time

B_4: Client Receive Response Time

End-to-end Latency = $B_4 - B_1$

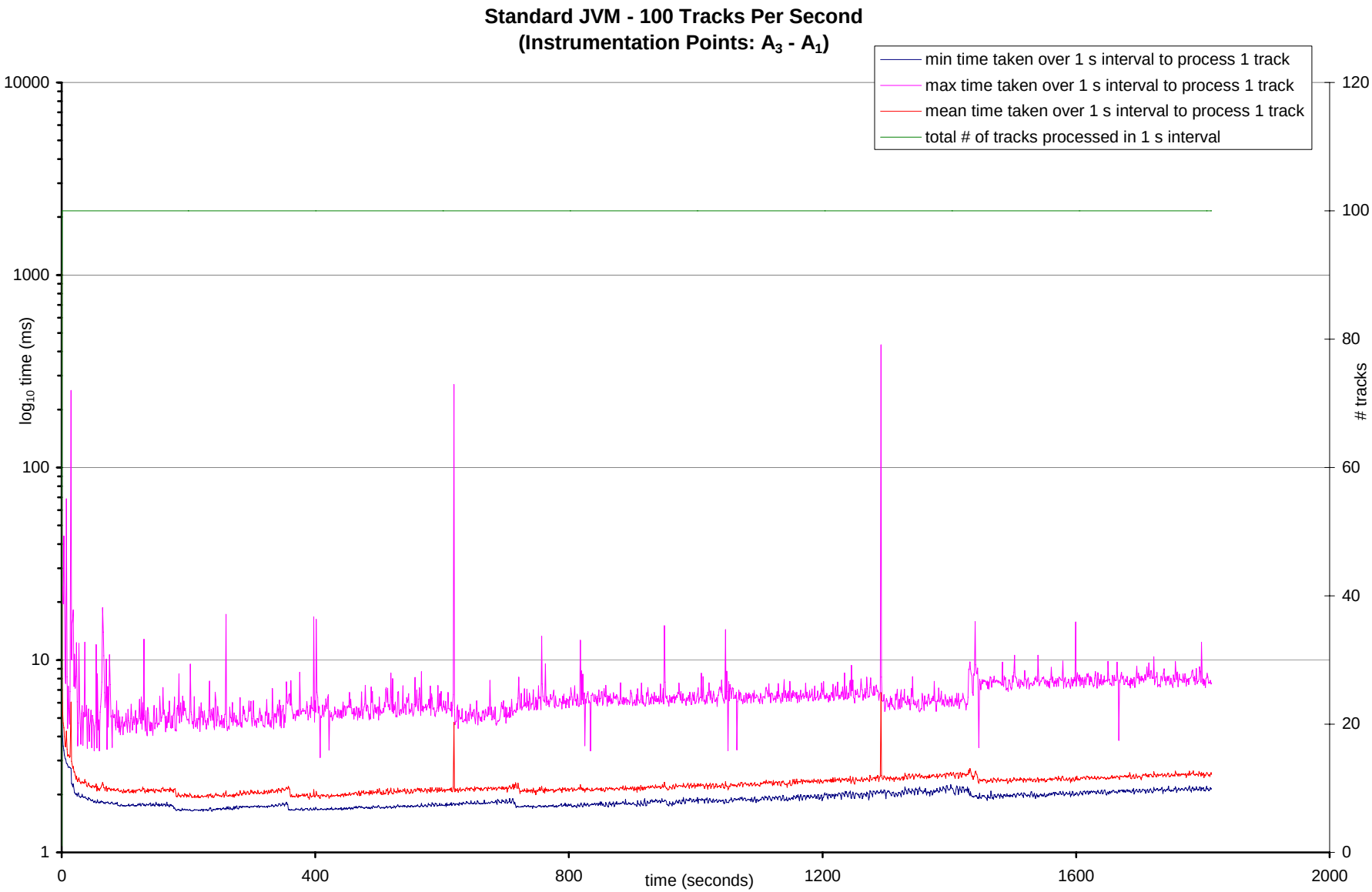
Service-to-client Latency = $B_4 - B_3$

Experiment Metrics

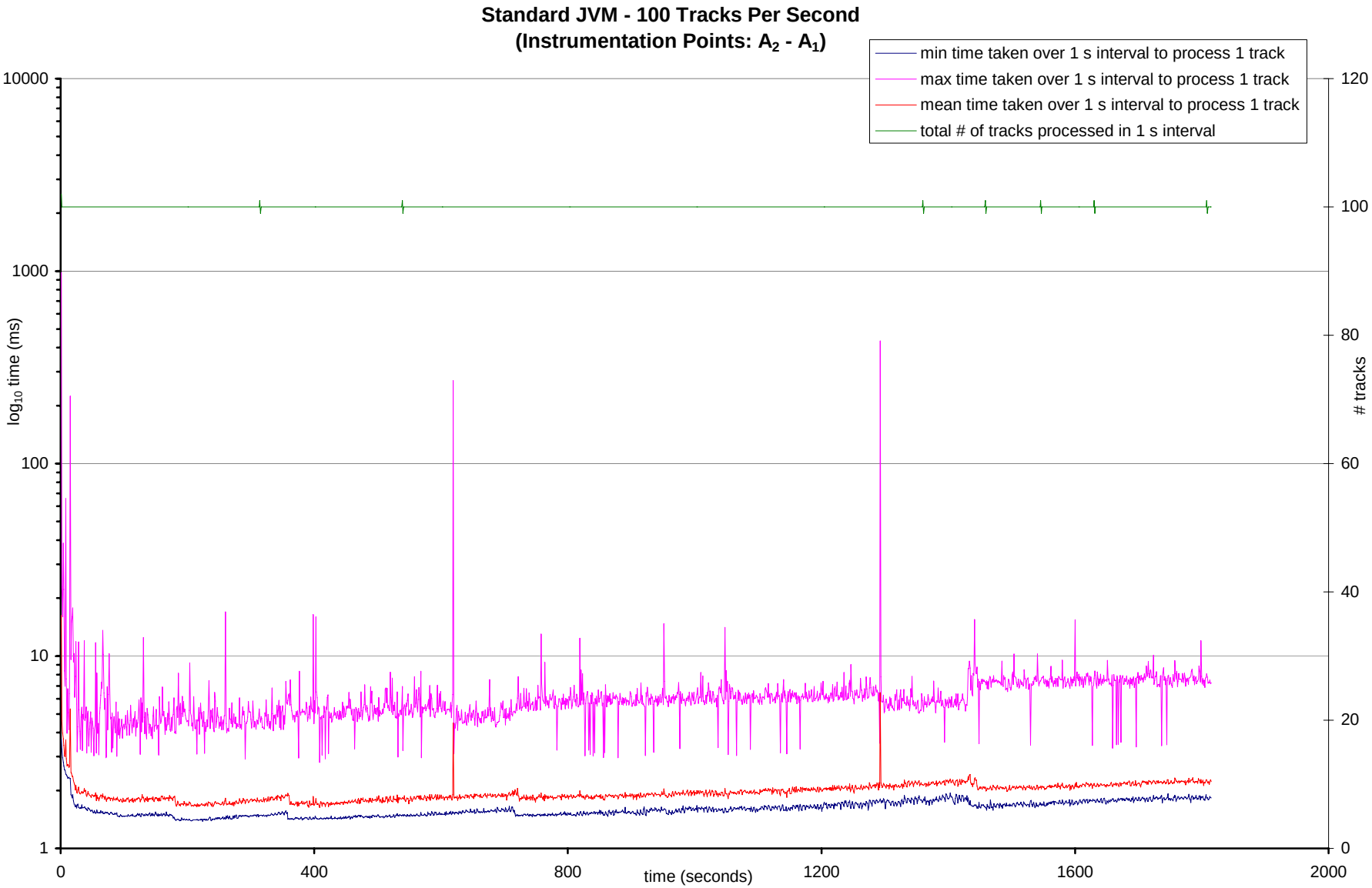
- Transport and processing latencies
  - Client Callback latencies
    - Min, max, and mean
  - Client-to-service latencies
    - Min, max, and mean
  - End-to-end latencies
    - Min, max, and mean

Results

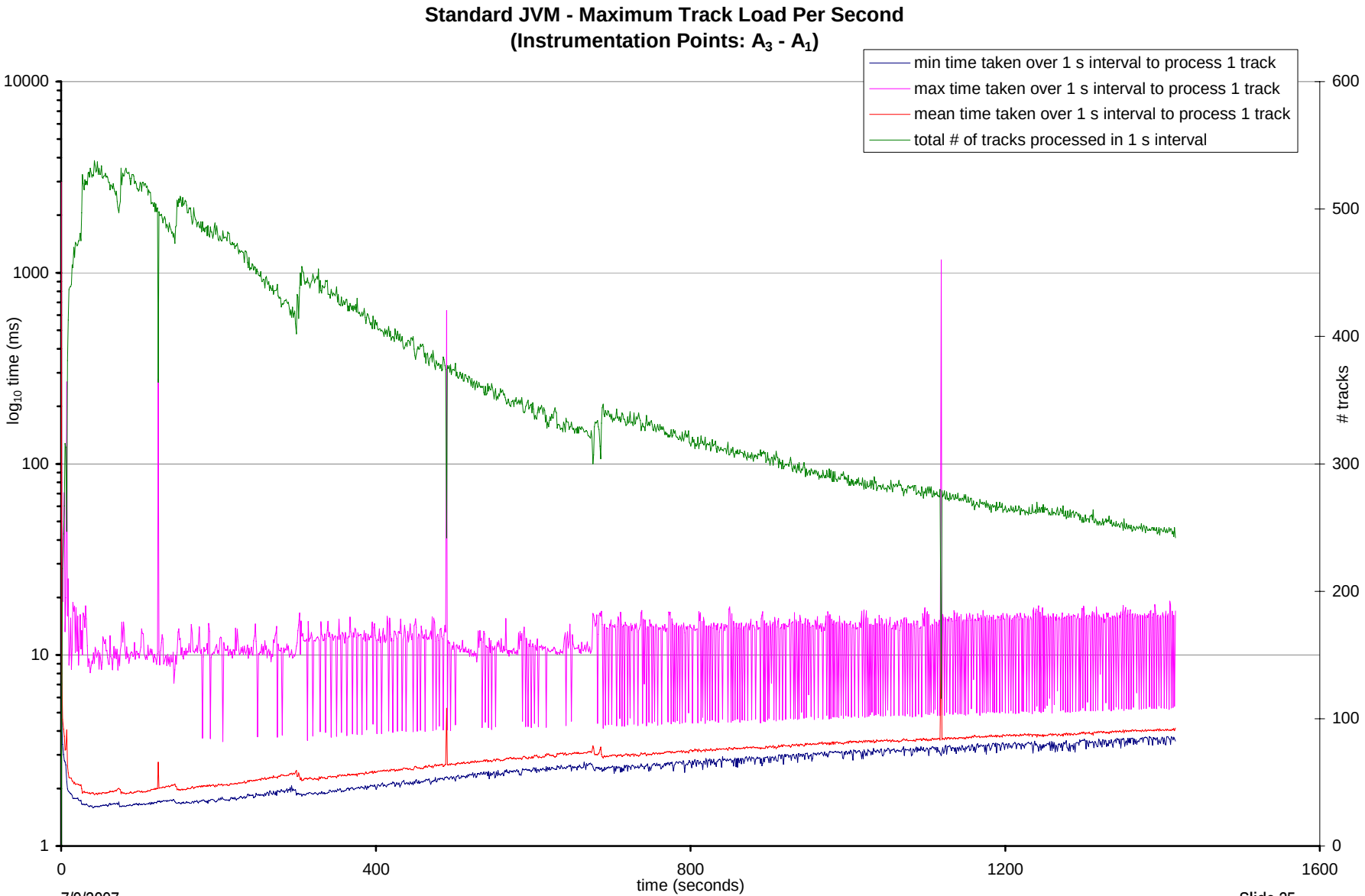
Scenario #1: Client Callback Latencies, Test 1



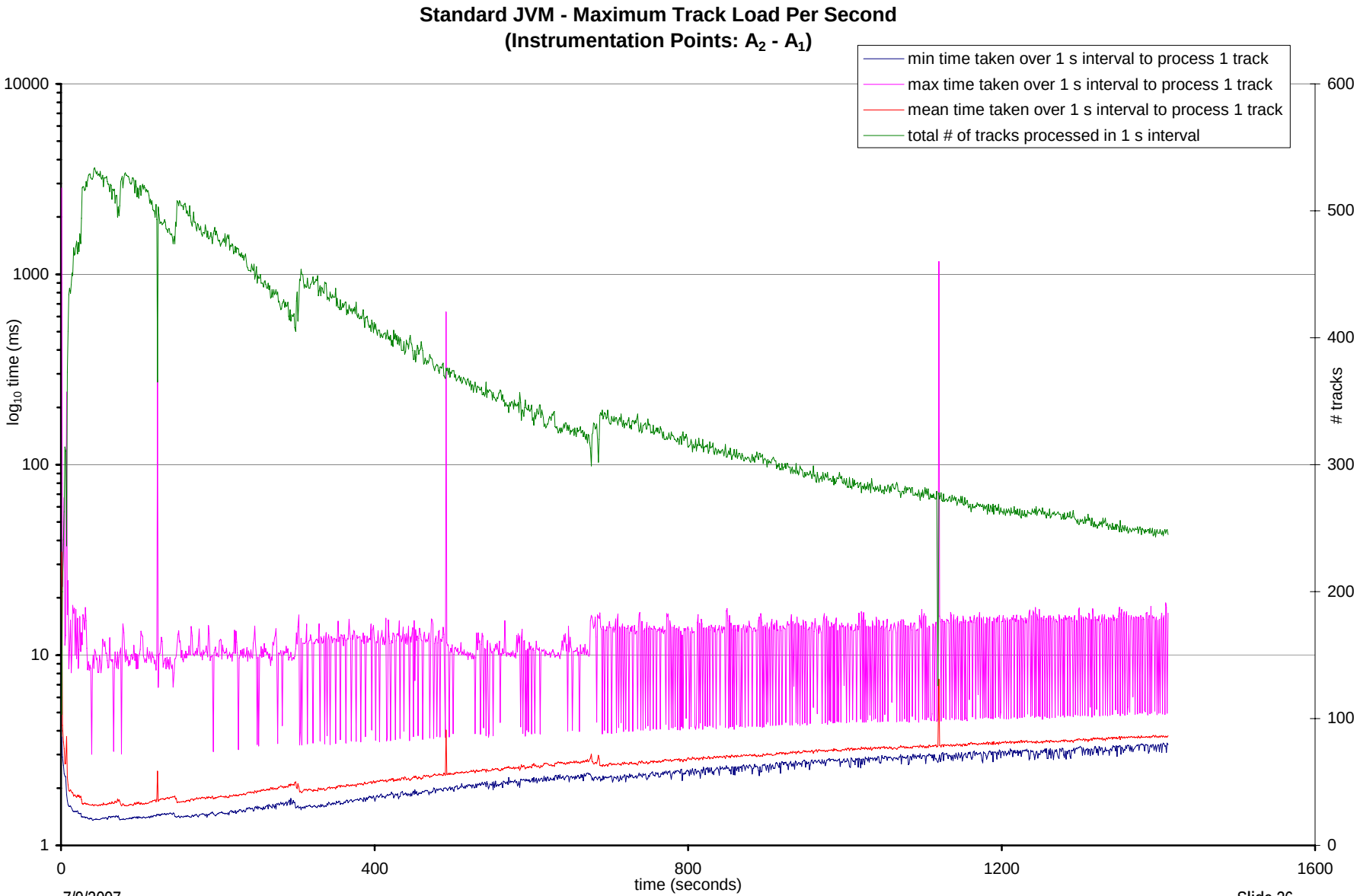
Scenario #1: Client-to-Service Latencies, Test 1



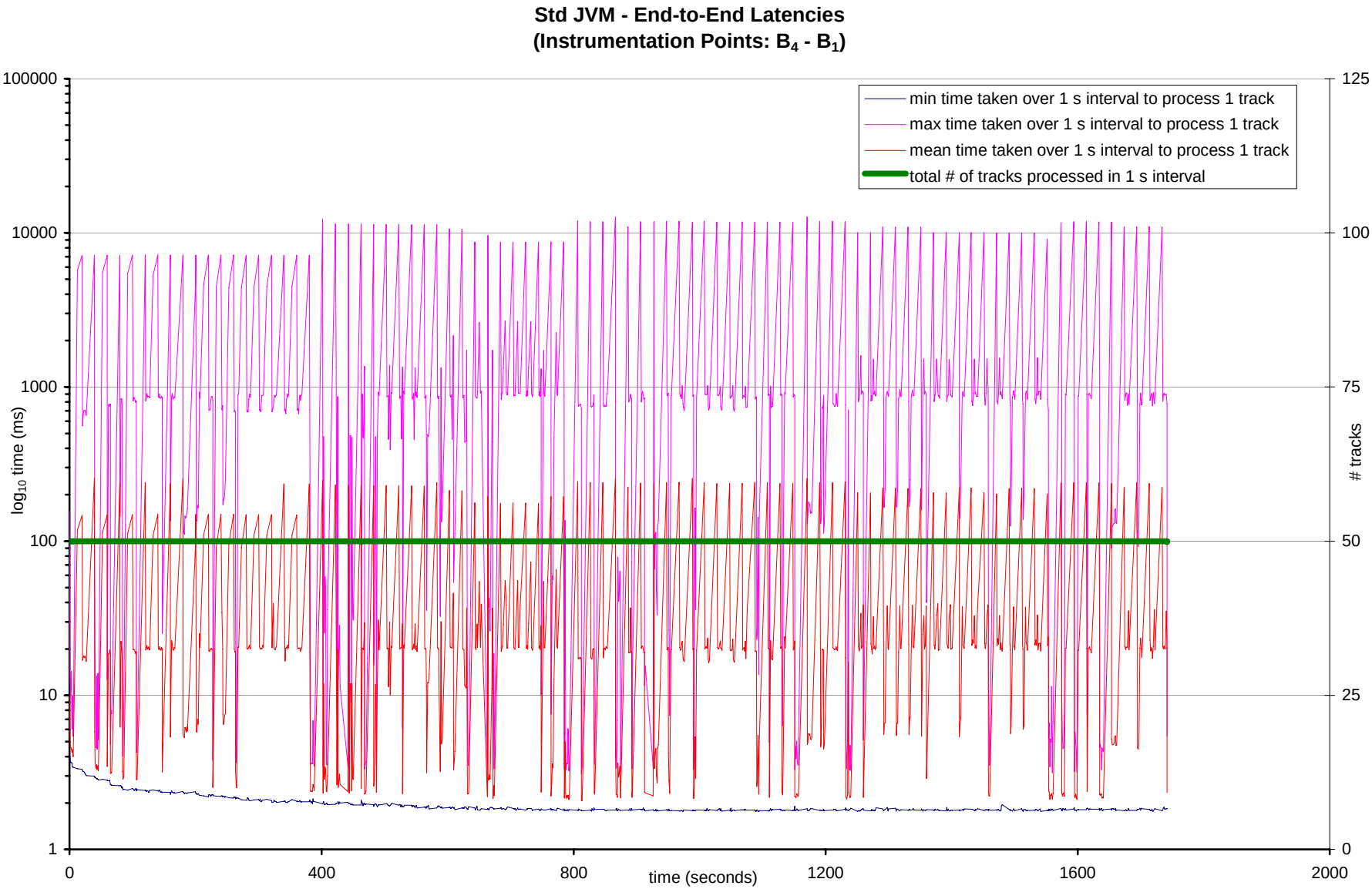
Scenario #1: Client Callback Latencies, Test 2



Scenario #1: Client-to-Service Latencies, Test 2



Scenario #2: End-to-End Latencies, Test 3



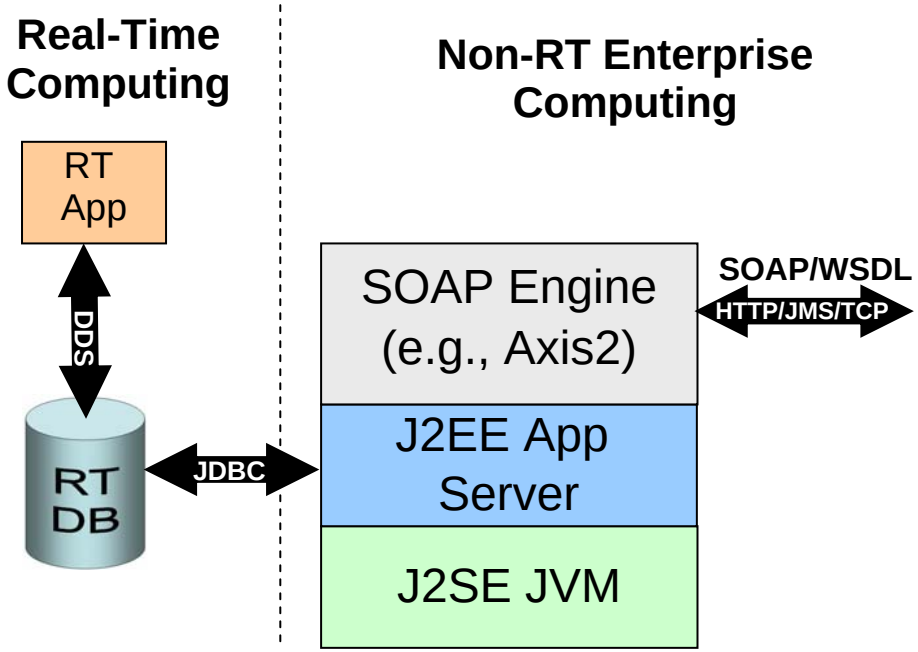
Conclusions

Conclusions

- Web Services technologies show great potential, but some barriers need to be addressed if they are to be integrated in RT naval combat system applications
- Developments in the technology base required for use in combat systems:
  - RT J2EE application servers
  - Binary XML standards and technology offerings – watching W3C efforts with interest
  - Closer integration of the Web Services with other transports besides HTTP
- Other aspects to be rigorously investigated include Web Services security, reliability, and pub/sub mechanisms

Can the Two Computing Domains be Merged?

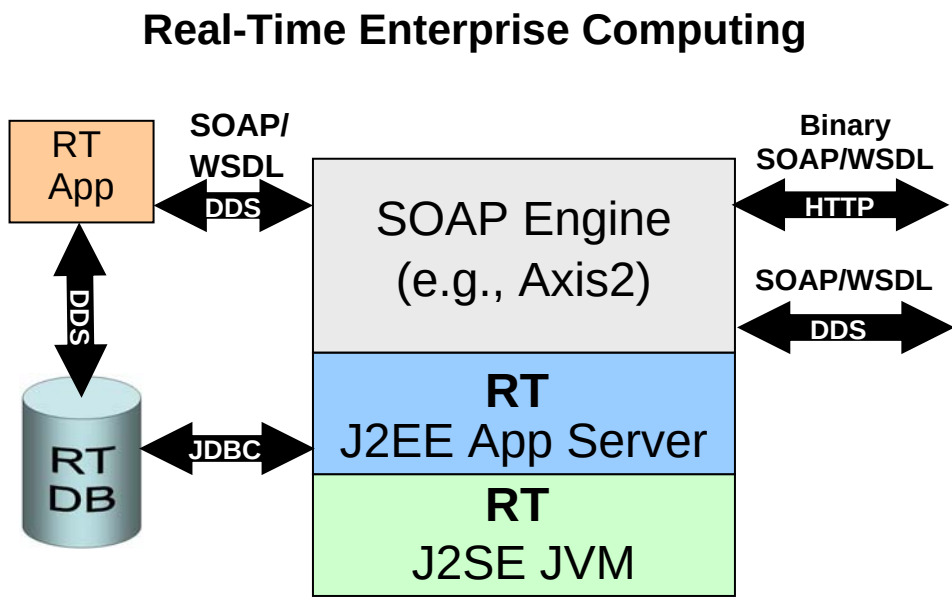
An Example Today



Current Limitations

- Enterprise computing infrastructures (J2EE) at the fringes of RT
- XML size and processing issues
- Transport latencies (HTTP/JMS/TCP)

Future Possibility?



Future Possibilities

- RT J2EE computing infrastructures
- XML over DDS or binary XML over HTTP
- DDS transport integrated w/ Enterprise infrastructures

Questions