OMG's Workshop on Distributed Object Computing for Real-Time and Embedded Systems

July 14-17, 2003 - Arlington, VA

Performance Testing of the TENA Middleware

Gregory L. Schultz

Speaker Introduction

- Member of the Test & Training Enabling Architecture (TENA) Architecture Management Team (AMT)
- Project Lead for one of the FI2010 Test Cases
- Development member on FI2010 Test Team

Presentation Goals

- Briefly describe the TENA middleware
- Discuss results of performance testing
- Enumerate benefits of using the TENA middleware

What is TENA?

- Means "Test & Training Enabling Architecture"
- Defines a common architecture for the test and training range community
- Provides a common software framework called the TENA middleware
- Uses the concept of a Logical Range - "a suite of TENA resources, sharing a common object model, that work together for a given range event"
- Defines a meta-model for the definition of Logical Range Object Models (LROMs)

What is TENA? (Continued)

- **Will eventually define common tools, concept of operations, and standard object models to facilitate multi-range exercises**

Stateful Distributed Object (SDO)

- **SDO classes specified using the TENA Definition Language (TDL)**
  - **Based on the OMG's Interface Definition Language (IDL) but with extensions to support the SDO concept**
- **The creator and maintainer (i.e. the "owner") of an SDO instance is called a *servant.***
  - **There is only one servant for any particular SDO instance**
- **The remote instance of the SDO is called a *proxy.***
  - **There can be one or more proxies for any SDO servant**

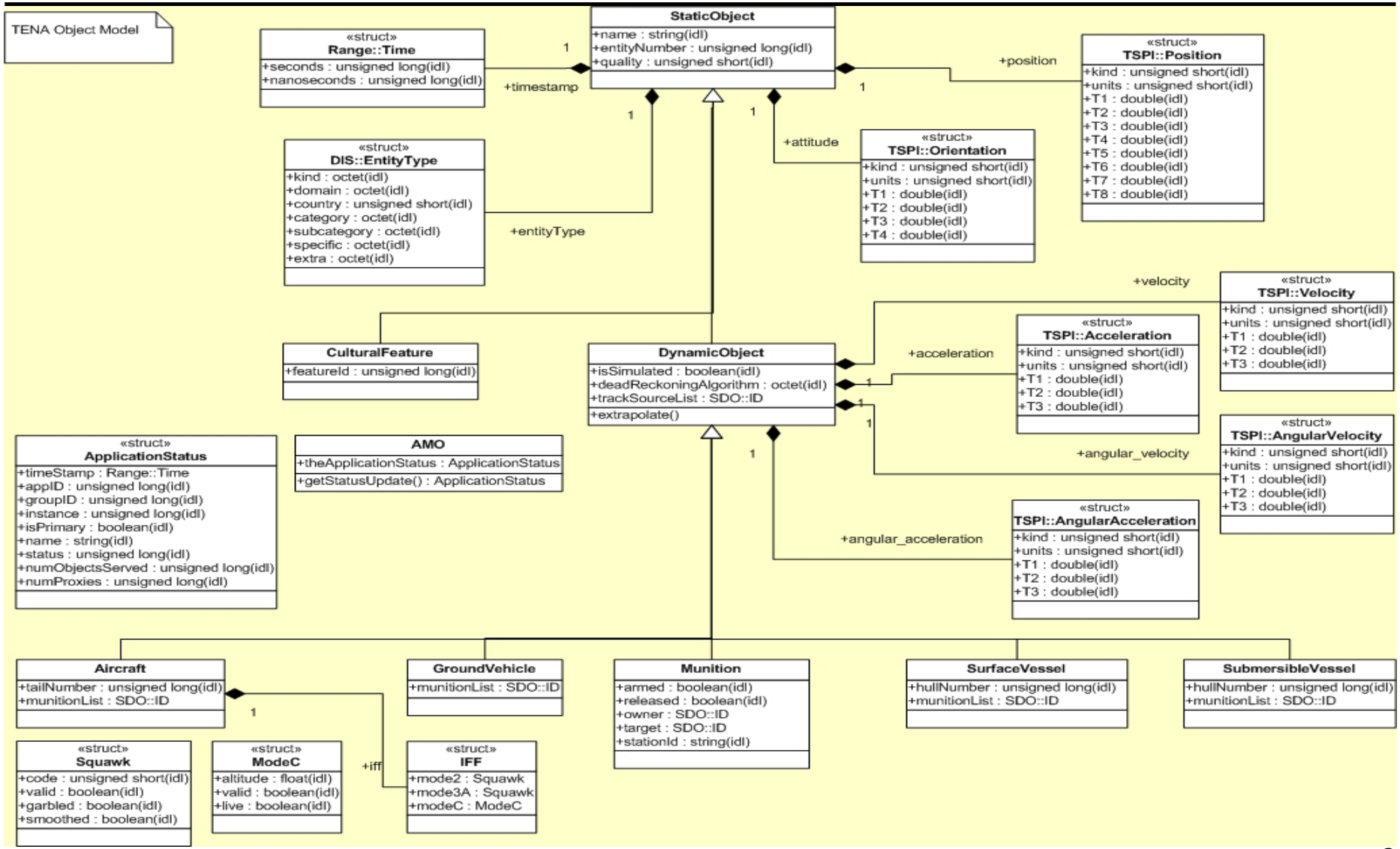
Stateful Distributed Objects (Continued)

- **Provides remotely-invocable methods**
  - **Implemented by servant in the server application**
  - **Invoked in client application by using proxy**
- **Provides publication state**
  - **Attributes automatically disseminated to all client applications that subscribe to a particular SDO**
  - **Can be considered a form of distributed shared memory**
- **Support inheritance and composition**
  - **can extend (“inherit from”) another SDO class**
  - **can contain (or be contained in) another SDO class**

Test Case Description

- Used Release 3.0 of the TENA middleware
- Developed Test Case Matrix which
  - published at the various levels of class inheritance
  - subscribed at the full published class level or at some base-class level
- Each test case varied publication state update rate and number of published SDO instances
  - at 1 Hz varied the published SDO instances from 1 - 256
  - at 10 Hz varied the published SDO instances from 1 - 64
  - at 100 Hz published only one SDO instance
- Compared TENA Middleware performance verses that of TAO Real-Time Event Service

TENA Object Model

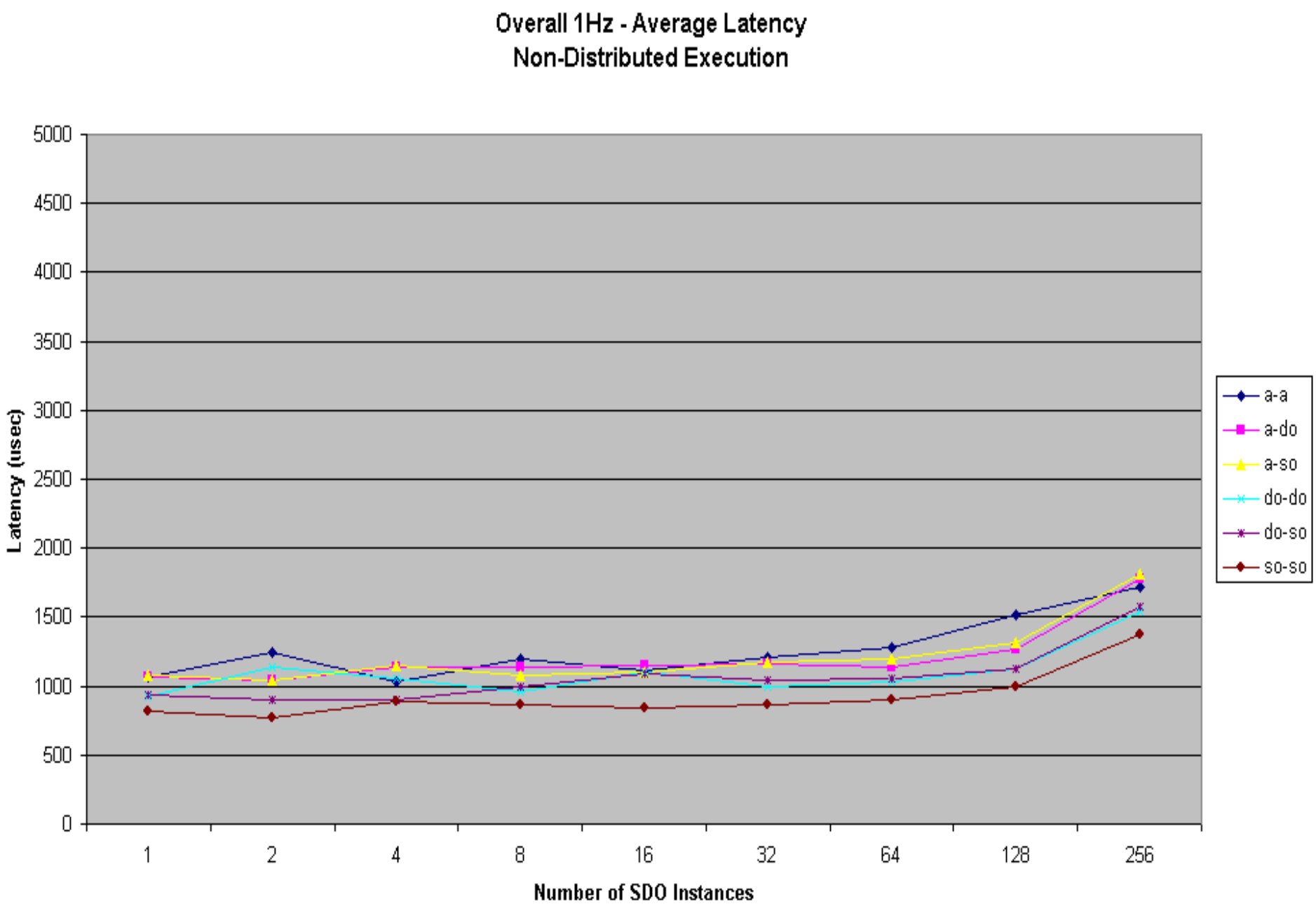


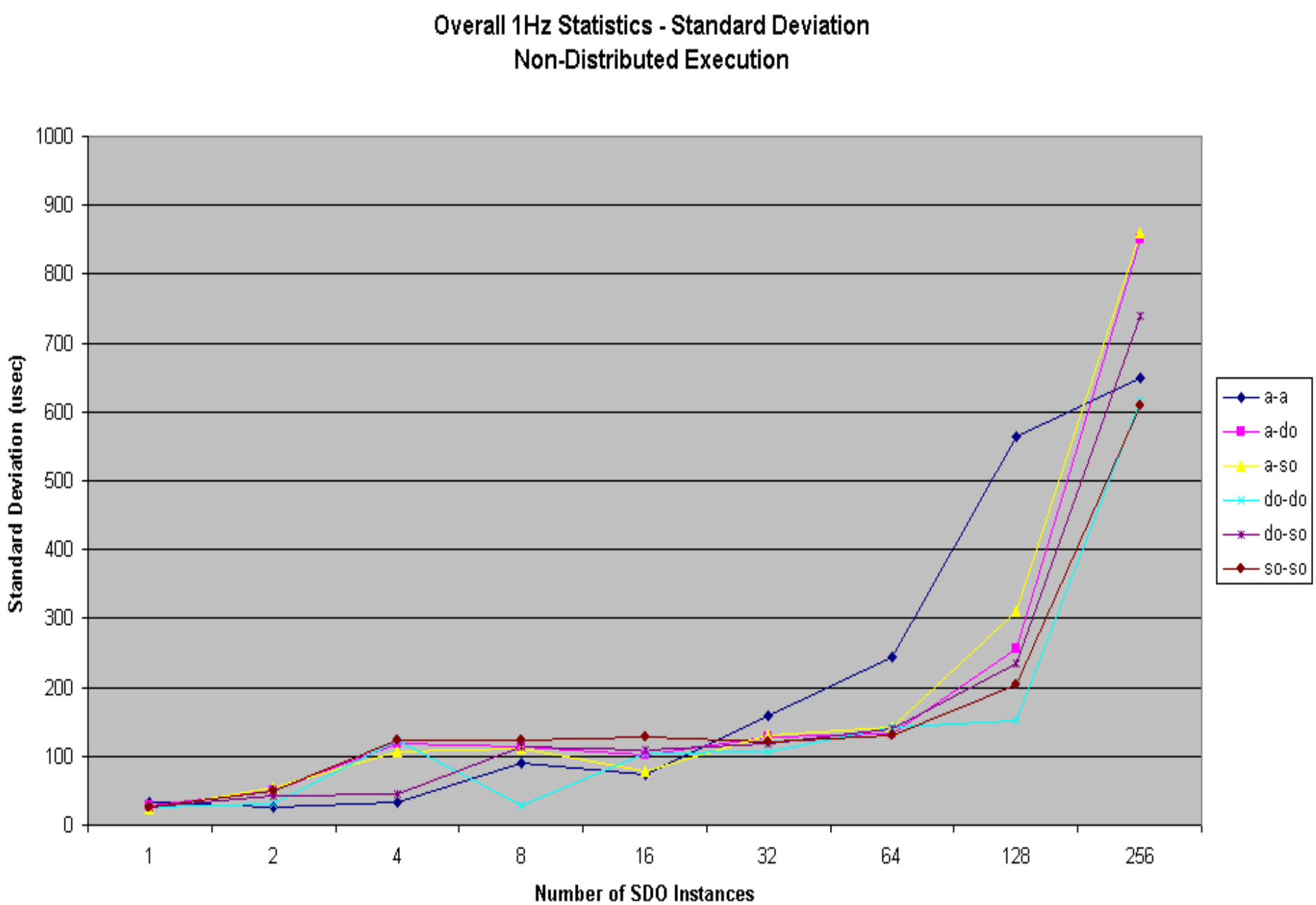
Test Case Matrix

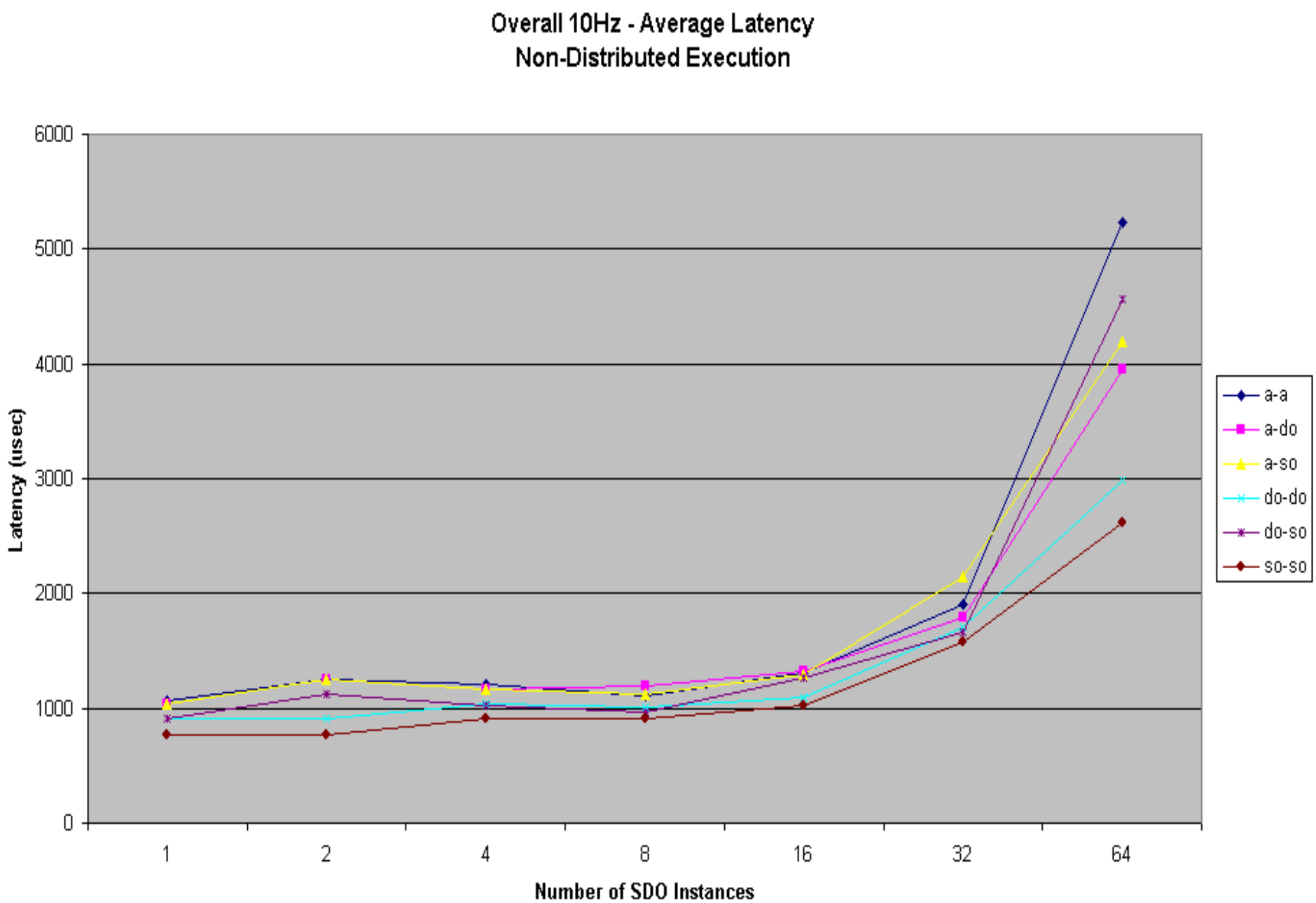
Test Case	Test ID	Aircraft SDO Publisher	Dynamic Object SDO Publisher	Static Object SDO Publisher	Aircraft SDO Subscriber	Dynamic Object SDO Subscriber	Static Object SDO Subscriber
1	A-SO	X					X
2	A-DO	X				X	
3	A-A	X			X		
4	DO-SO		X				X
5	DO-DO		X			X	
6	SO-SO			X			X

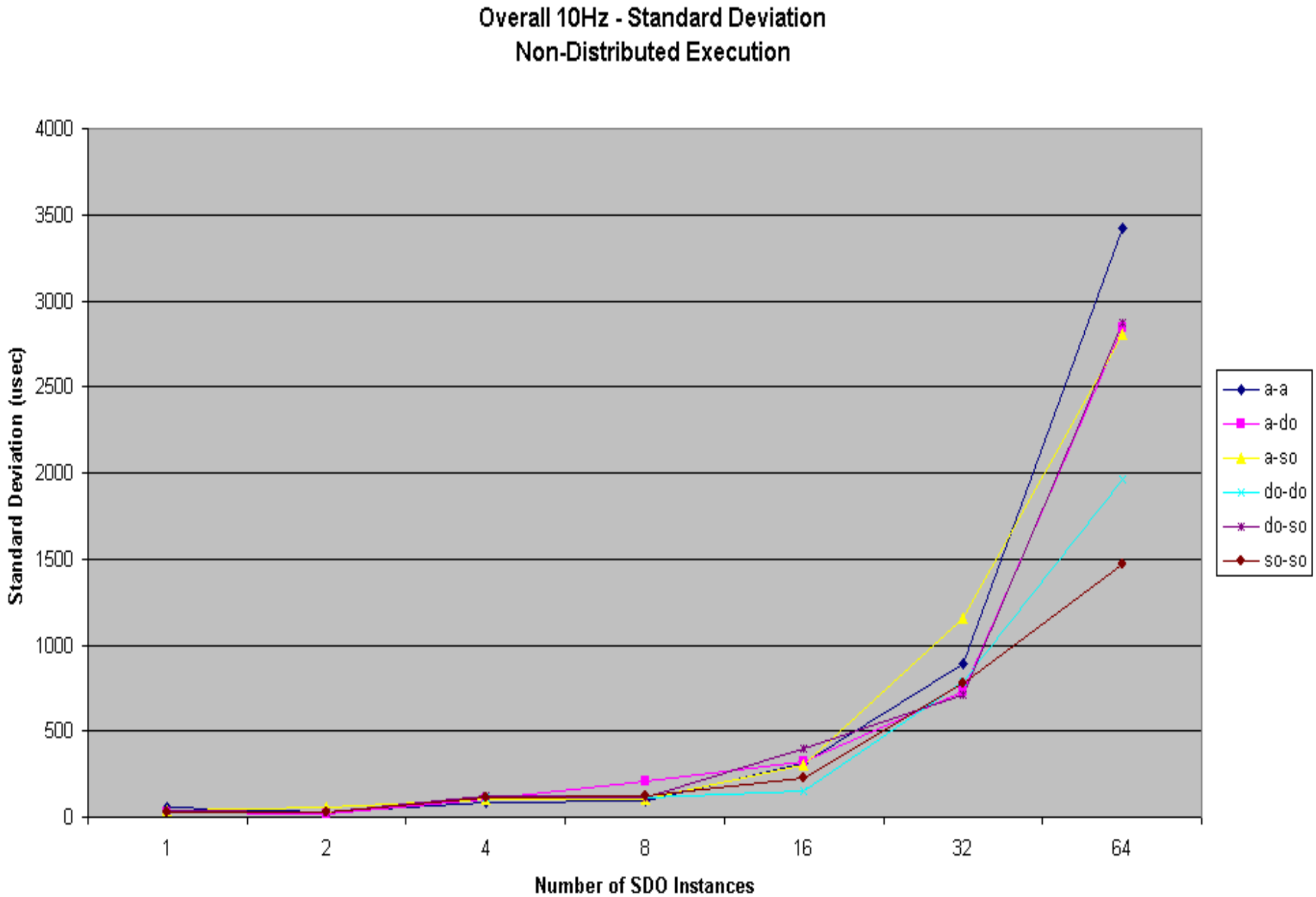
Test Platform Description

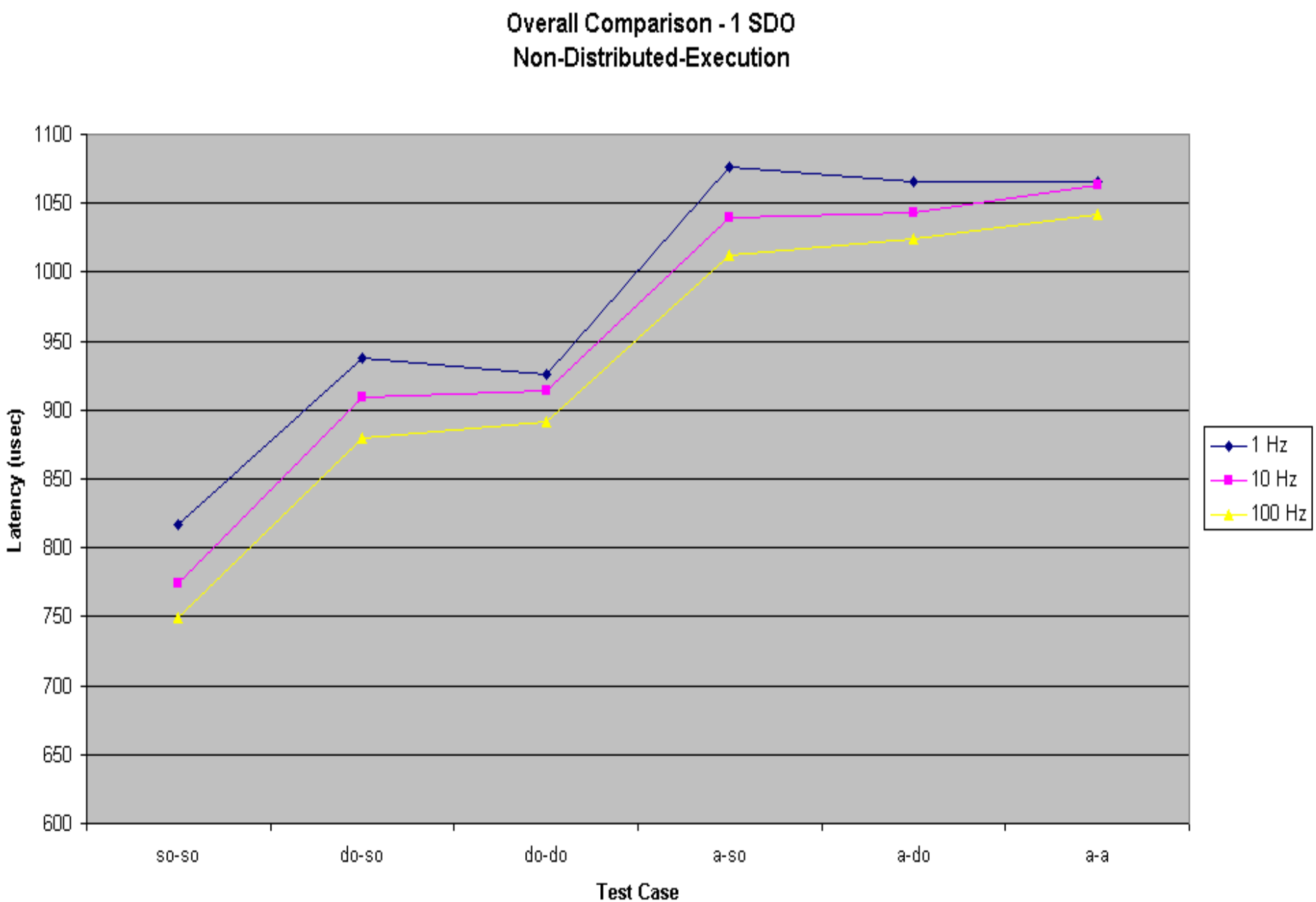
Computer System Characteristic	System 1
Name	localhost
System Manufacturer	IBM PC compatible
System Model	n/a
CPU Type	1000MHz
Number of CPUs	2
CPU Cache	16K L1, 256K L2
RAM	512 MB
RAM Speed	unknown
System Bus Speed	133MHz FSB
Network Interface Type (ATM, Ethernet, Scramnet, etc.)	Ethernet
NIC Speed (Bits/s)	100 MB
NIC Manufacturer	3Com
NIC Model/Type	3c905C-TX
Operating System	Linux (RedHat 7.1)
OS Version/Release	Linux 2.4.2-2smp (SMP kernel)


Overall 1Hz - Average Latency
Non-Distributed Execution
Latency (usec)
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1
2
4
8
16
32
64
128
256
Number of SDO Instances
a-a
a-do
a-so
do-do
do-so
so-so


Overall 1Hz Statistics - Standard Deviation
Non-Distributed Execution
Standard Deviation (usec)
1000
900
800
700
600
500
400
300
200
100
0
1
2
4
8
16
32
64
128
256
Number of SDO Instances
a-a
a-do
a-so
do-do
do-so
so-so


Overall 10Hz - Average Latency
Non-Distributed Execution
Latency (usec)
6000
5000
4000
3000
2000
1000
0
1
2
4
8
16
32
64
Number of SDO Instances
a-a
a-do
a-so
do-do
do-so
so-so

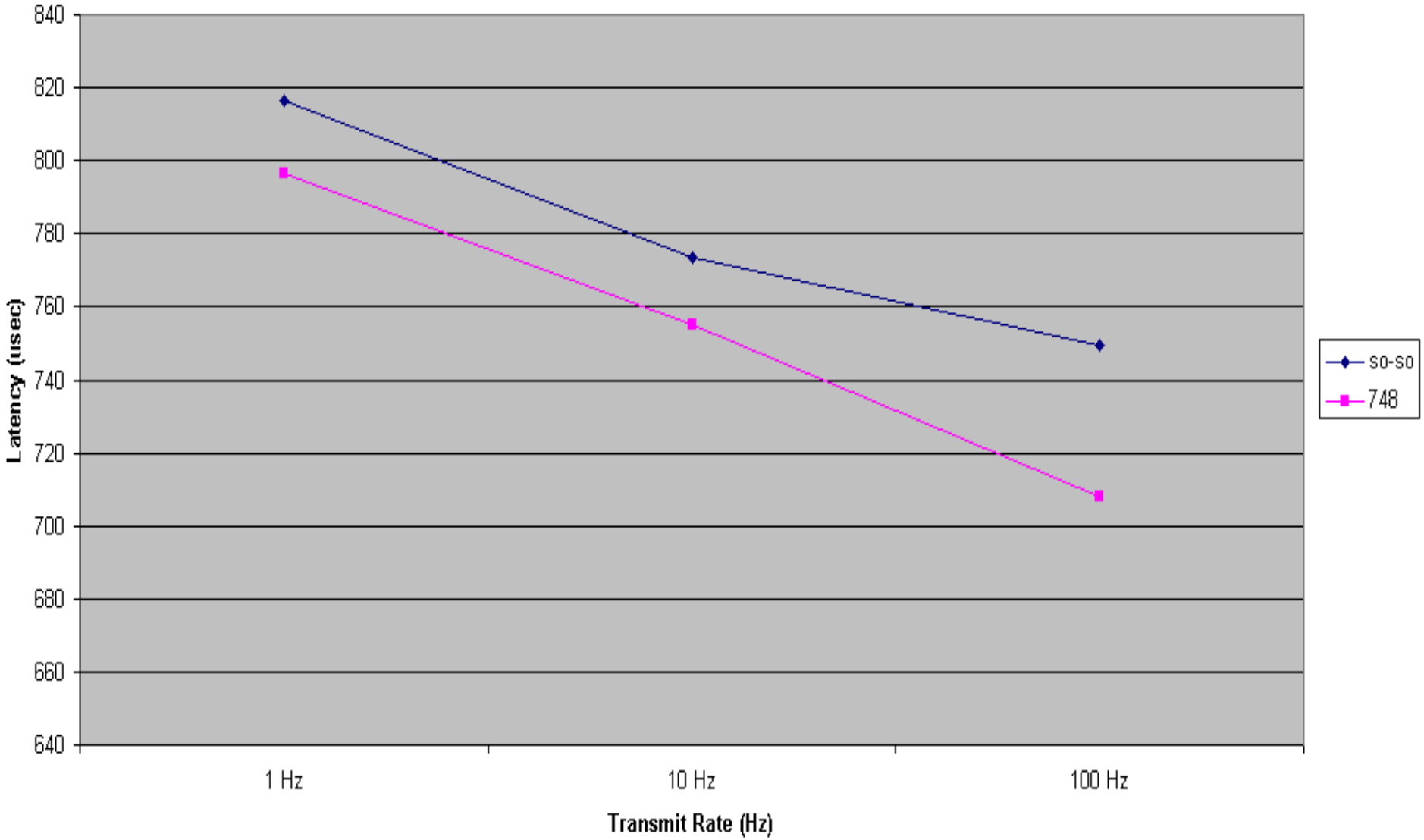

Overall 10Hz - Standard Deviation
Non-Distributed Execution
Standard Deviation (usec)
4000
3500
3000
2500
2000
1500
1000
500
0
1
2
4
8
16
32
64
Number of SDO Instances
a-a
a-do
a-so
do-do
do-so
so-so

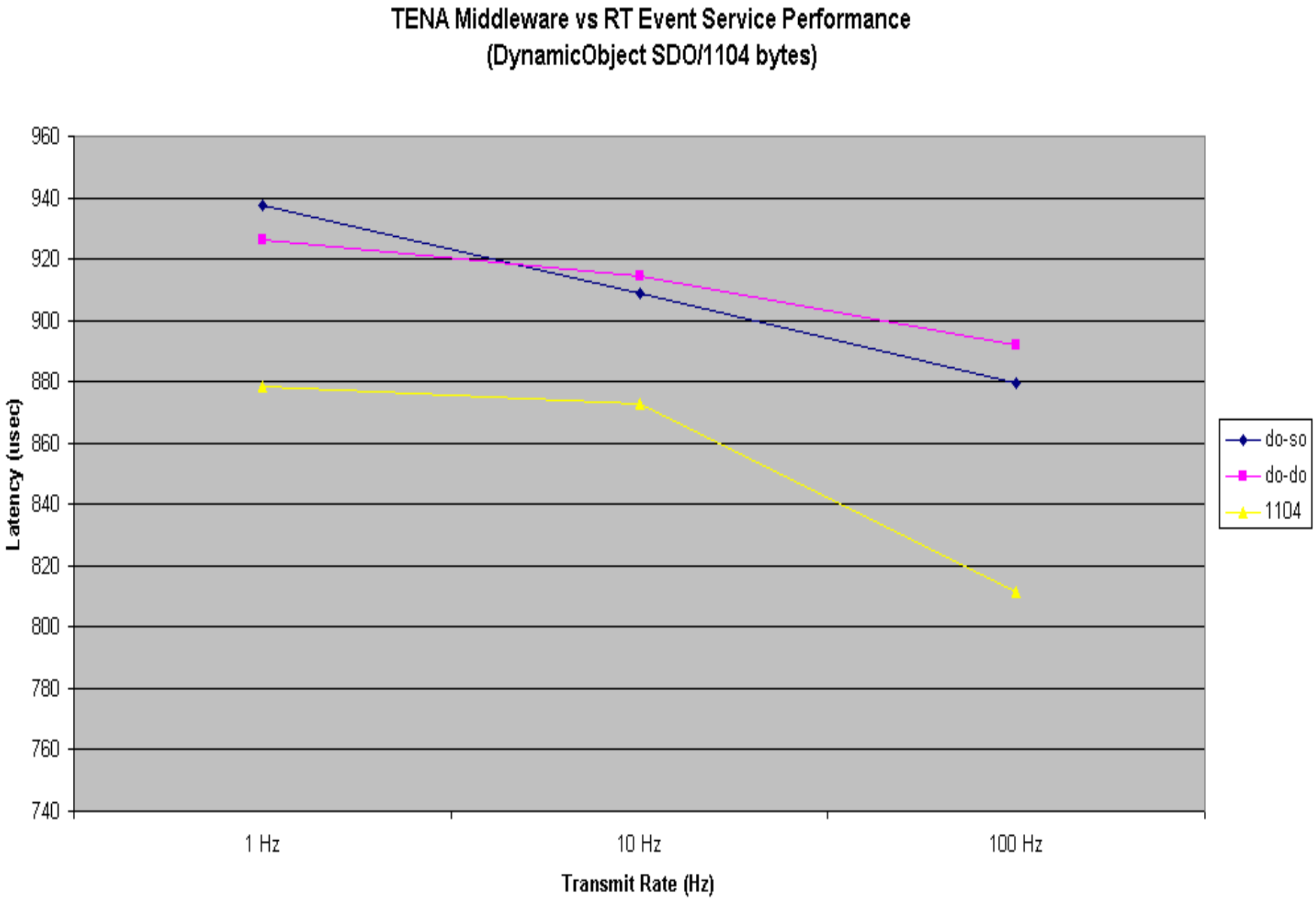

Overall Comparison - 1 SDO
Non-Distributed-Execution
Latency (usec)
1100
1050
1000
950
900
850
800
750
700
650
600
so-so
do-so
do-do
a-so
a-do
a-a
Test Case
1 Hz
10 Hz
100 Hz

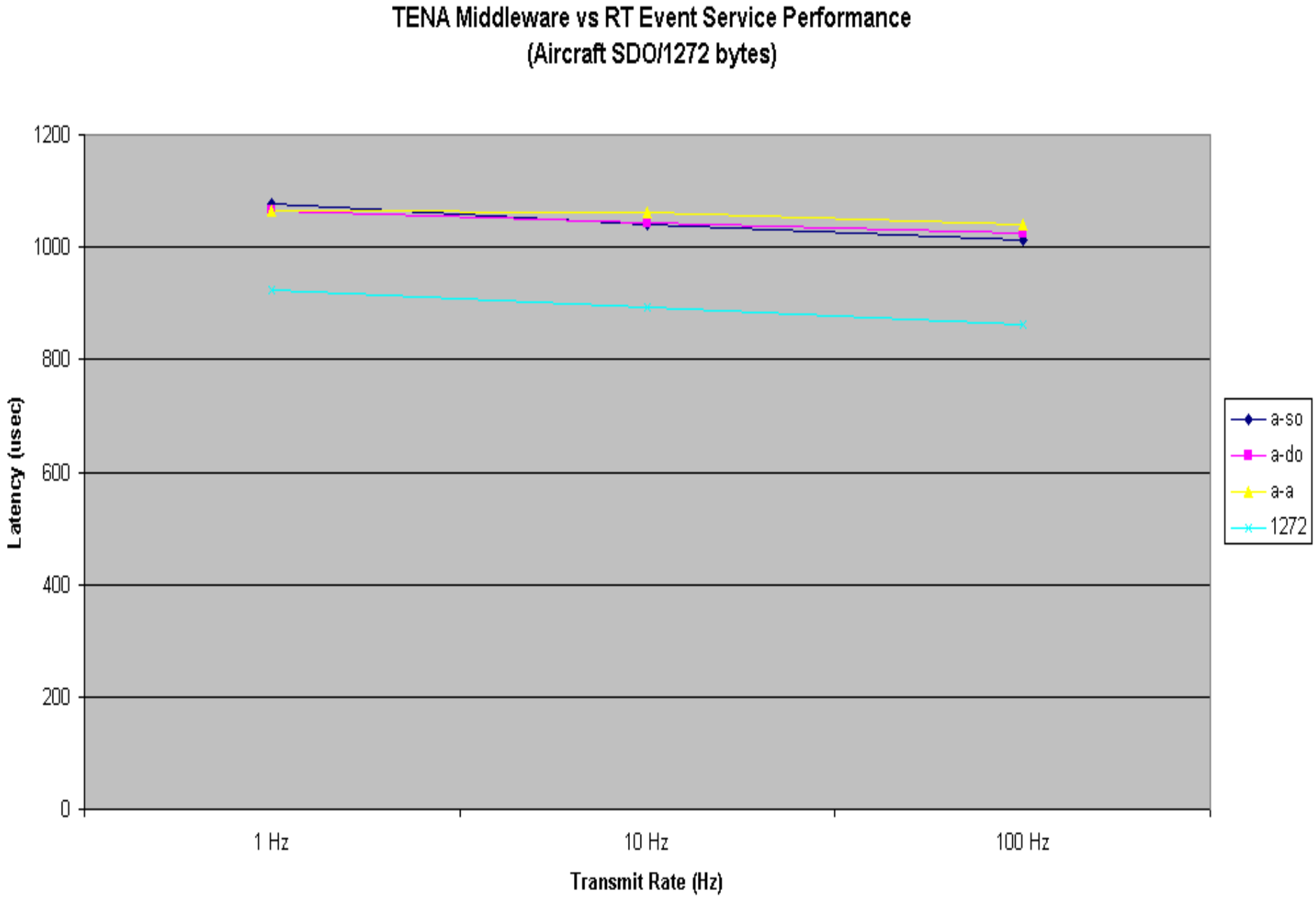
TENA Object Model
SDO Publication State Packet Sizes

SDO	Packet Size
StaticObject	748 bytes
DynamicObject	1104 bytes
Aircraft	1272 bytes

TENA Middleware vs RT Event Service Performance
(StaticObject SDO/748 bytes)




TENA Middleware vs RT Event Service Performance
(DynamicObject SDO/1104 bytes)
960
940
920
900
880
860
840
820
800
780
760
740
Latency (usec)
1 Hz
10 Hz
100 Hz
Transmit Rate (Hz)
do-so
do-do
1104


TENA Middleware vs RT Event Service Performance
(Aircraft SDO/1272 bytes)
1200
1000
800
600
400
200
0
Latency (usec)
1 Hz
10 Hz
100 Hz
Transmit Rate (Hz)
a-so
a-do
a-a
1272

Publisher Example Source-Code

```
#include <TENA/IKE2/IKE2.h>
#include "TP_Task.h"

int main(int argc, char * argv[] )
{
  ...

  TENA::Middleware::Configuration config( argc, argv );

  TENA::Middleware::RuntimePtr pRuntime = TENA::Middleware::init( config );

  TENA::Middleware::ExecutionPtr pExecution = pRuntime->joinExecution( executionName );
  ...

for (int i=0; i<(numberInstances+perThread-1)/perThread; i++)
    {
      TP_Task *updaterThread = new TP_Task(
        pExecution, sessionName,
        (commTypeStr == "BestEffort") ? 1 : 0, theDuration, i*perThread,
        (i*perThread+(perThread-1) < numberInstances) ? i*perThread+(perThread-1)
                                                     : numberInstances-1 );
      if (updaterThread != NULL)
        updaterThread->activate( THR_NEW_LWP, 1, 0, ACE_DEFAULT_THREAD_PRIORITY );
    }
  ...
  return 0;
}
```

Publisher Example Source-Code (Continued)

```
#include <time.h>
#include <map>
#include <memory>

#include "ace/Task.h"
#include "tao/corba.h"

#include <TENA/IKE2/IKE2.h>
#include <OMstd/StaticObject/StaticObjectPublisher.h>
#include "StaticObjectRemoteServantMethodsFactoryImpl.h"
#include <strstream>

#include "Timer.h"

class TP_Task : virtual public ACE_Task_Base
{
private:
  TENA::Middleware::SessionPtr                                  pSession_;
  std::auto_ptr< OMstd::StaticObject::ServantFactory_t >        servantFactory_;
  unsigned long long                                            duration_;
  int                                                           done_;
  int                                                           startInstance_;
  int                                                           endInstance_;
  std::map< std::string, OMstd::StaticObject::ServantPtr >      objectMap_;

public:
```

Publisher Example Source-Code (Continued)

```
TP_Task(TENA::Middleware::ExecutionPtr pExecution, std::string sessionName, int commType,
          unsigned long long duration, int startInstance, int endInstance) :
     duration_(duration), done_(0),
     startInstance_(startInstance), endInstance_(endInstance)
  {
    pSession_ = pExecution->createSession( sessionName );

    OMstd::StaticObject::RemoteServantMethodsFactoryPtr
      pDefaultStaticObjectRemoteServantMethodsFactory(
        new OMimpl::StaticObject::StaticObjectRemoteServantMethodsFactoryImpl() );

    servantFactory_ =
        IKE2::publish< OMstd::StaticObject::ServantTraits >(
          pSession_,
          pDefaultStaticObjectRemoteServantMethodsFactory );

    for (int i=startInstance; i <= endInstance_; i++)
      {
        std::ostrstream ostr;
        ostr << "SlamDunk" << i << ends;

        objectMap_[ostr.str( )] = servantFactory_->createServantUsingDefaultFactory(
          (commType == 1) ? TENA::Middleware::BestEffort : TENA::Middleware::Reliable );

        std::auto_ptr< OMstd::StaticObject::PublicationStateUpdater > pUpdater(
          objectMap_[ostr.str( )]->createPublicationStateUpdater() );
```

Publisher Example Source-Code (Continued)

```
      OMstd::Range::Time  timestamp;

      pUpdater->setName( ostr.str( ) );

      ACE_Time_Value tv = Timer::updateTime( );
      timestamp.seconds( tv.sec() );
      timestamp.nanoseconds( tv.usec() );
      pUpdater->setTimestamp( timestamp );

      objectMap_[ostr.str( )]->modifyPublicationState( pUpdater );
    }
}
```

Publisher Example Source-Code (Continued)

```
virtual int svc()
  {
    struct timespec tm;
    ::ACE_hrtime_t shrtime, ehrtime, nhrtime;
    unsigned long long deltaTime;

    for (;done_ == 0;)
      {
        shrtime = ACE_OS::gethrtime( );
        for (int i=startInstance_; i <= endInstance_; i++)
          {
            std::ostrstream ostr;
            ostr << "SlamDunk" << i << ends;

            std::auto_ptr< OMstd::StaticObject::PublicationStateUpdater > pUpdater(
              objectMap_[ostr.str( )]->createPublicationStateUpdater() );

            OMstd::Range::Time  timestamp;

            pUpdater->setName( ostr.str( ) );

            ACE_Time_Value tv = Timer::updateTime( );
            timestamp.seconds( tv.sec() );
            timestamp.nanoseconds( tv.usec() );
            pUpdater->setTimestamp( timestamp );

            objectMap_[ostr.str( )]->modifyPublicationState( pUpdater );
          }
```

Publisher Example Source-Code (Continued)

```
        if (duration_ < 10000000)
        {
          nhrtime = shrtime + duration_;
          do {
            ehrtime = ACE_OS::gethrtime( );
          } while (ehrtime < nhrtime);
        }
        else
        {
          nhrtime = ACE_OS::gethrtime( );
          deltaTime = nhrtime - shrtime;
          if (deltaTime > duration_) continue;
          deltaTime = duration_ - deltaTime;
          tm.tv_sec  = deltaTime / 1000000000UL;
          tm.tv_nsec = deltaTime - tm.tv_sec * 1000000000UL;
          ACE_OS::nanosleep( &tm, NULL );
        }

      }

    return 0;
  }

};
```

Conclusion

Benefits of The TENA Middleware

- **Abstracts the real-world domain into objects**
  - **Promotes object-oriented analysis, design and programming practices**
- **Programmer does not have to deal with the low-level socket API**
  - **TENA Middleware (via TAO CORBA ORB) manages the connections**
  - **Programmer can specify type of connection per published object - reliable (TCP/IP) or best-effort (multicast UDP)**
- **Increases programmer productivity and reduces errors**
  - **TENA uses UML-based model-driven automated code generation**
  - **TENA API provides compile-time type-safety**

Benefits of The TENA Middleware (Continued)

- Implements a publish/subscribe communication paradigm
  - Similar to HLA (High Level Architecture)
- Implements remote method invocation
  - Similar to CORBA
- Promotes fault tolerant software systems
  - Publishers and subscribers can exit exercise at any time without effecting each other
- Middleware distributions for UNIX (Linux/Solaris) and Windows (NT/2000/XP) platforms
  - IRIX and VxWorks distributions possible in near future

Questions or Comments?

- **Phone: (850) 682-1184**
- **E-Mail: gregory.schultz@warpcomputing.com or gregory.schultz@cox.net**