Open Architecture Publish-Subscribe Benchmarking

Bruce McCormick
bruce.mccormick.ctr@navy.mil
Objectives

- Understand ability of products to support bounded latencies and sustained throughput required by combat system applications.
- Understand architectural differences in products that may affect ability of products to scale to large numbers of processors and processes.
- Provide a baseline of comparison as products implement DDS compliance.
- Understand performance impacts of DDS Entity QoS settings.
- Understand impacts of communications system design.
- Understand discovery and system metadata mechanisms and performance for various network topologies.
Performance Considerations

Focus is to rapidly characterize various system elements affecting performance

- Overall communication system design
  - Readers / writers / rates / sizes / types / topics / domains / partitions
  - Fault tolerance - redundancy

- Application design
  - Instances / samples
  - Waitsets / listeners

- Resource allocations exposed through QoS
  - System
  - Transport
  - Host

- Real Time requirements
  - Highly predictable, distributed, asynchronous
  - Based on resource allocation

Implementation differences even as APIs converge
OpenSplice v2.0 beta

Node

Application

Listener/callback

DDS API

Pub/Sub M/W

Data Store

SPLICE Daemon

Network

Pub/Sub M/W

queries

wait for data

waitset

Pub

sub

 DDS API

Publisher/Callback

Publisher/Callback
Node

Application

publication

subscription

DDS API

Publish/Subscribe M/W

Application

waitsets

publication

subscription

queries

DDS API

Publish/Subscribe M/W

Network

RTI NDDS 4.1b beta
No problems with Integration of DDS
• NTP for clock sync
• NDDS 4.1b configuration
• OpenSplice 2.0 beta
  OpenFusion
• Vendor tool code generation
• C++ and Java
Entities

- All communication objects are specializations of abstract class Entity
  - Individual QoS policies
  - Individual communication status
- QoS of lower level entities take precedence over higher

<table>
<thead>
<tr>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>no attributes</td>
</tr>
<tr>
<td>operations</td>
</tr>
<tr>
<td>abstract_set_qos</td>
</tr>
<tr>
<td>qos_list</td>
</tr>
<tr>
<td>abstract_get_qos</td>
</tr>
<tr>
<td>abstract_set_listener</td>
</tr>
<tr>
<td>a_listener</td>
</tr>
<tr>
<td>mask</td>
</tr>
<tr>
<td>abstract_get_listener</td>
</tr>
<tr>
<td>get_statuscondition</td>
</tr>
<tr>
<td>get_status_changes</td>
</tr>
<tr>
<td>enable</td>
</tr>
<tr>
<td>get_instance_handle</td>
</tr>
</tbody>
</table>
Complexities

- Different entities and levels
  - *Topic, DataWriter, DataReader, Publisher, Subscriber, DomainParticipant.*

- Relationship between DDS QoS, node resources and infrastructure QoS

- Differences in feedback provided by the service when settings are unmet
  - RxO handles some aspects with notifications
    - Durability, presentation, deadline, ownership, latency budget, liveliness, reliability, destination order
  - Not others
    - History kind & depth (not to be confused with Durability Service History), transport priority, resource limits, lifecycles

- Example - Loss of reliable issues due to lack of resources
All Entities support listener access to status changes

Read communications (user data)

Plain communications (all others)
  - Lost / rejected samples
  - Discovery / liveliness
  - Incompatible QoS / topic consistency
  - Counts

Instrument provided callbacks with timestamps and write to METT
Available Status

- Status
  - SampleLostStatus
    - total_count: long
    - total_count_change: long
  - InconsistentTopicStatus
    - total_count: long
    - total_count_change: long
  - SampleRejectedStatus
    - total_count: long
    - total_count_change: long
    - last_reason: SampleRejectedStatus.Kind
    - last_instance_handle: InstanceHandle_t
  - PublicationMatchedStatus
    - total_count: long
    - total_count_change: long
    - current_count: long
    - current_count_change: long
    - last_subscription_handle: InstanceHandle_t
  - LivelinessChangedStatus
    - alive_count: long
    - not_alive_count: long
    - alive_count_change: long
    - not_alive_count_change: long
    - last_publication_handle: InstanceHandle_t
  - OfferedDeadlineMissedStatus
    - total_count: long
    - total_count_change: long
    - last_policy_id: QosPolicyId_t
    - policies [*]: QosPolicyCount
  - RequestedDeadlineMissedStatus
    - total_count: long
    - total_count_change: long
    - last_policy_id: QosPolicyId_t
    - policies [*]: QosPolicyCount
  - OfferedIncompatibleQosStatus
    - total_count: long
    - total_count_change: long
    - last_policy_id: QosPolicyId_t
    - policies [*]: QosPolicyCount
  - RequestedIncompatibleQosStatus
    - total_count: long
    - total_count_change: long
    - last_policy_id: QosPolicyId_t
    - policies [*]: QosPolicyCount
  - QosPolicyCount
    - policy_id: QosPolicyId_t
    - count: long
Built-In Topics

- DCPS specification introduces a set of built-in topics and corresponding DataReader objects that can then be used by the application. The information is then accessed as if it was normal application data.

- Notifies application of any changes by means of Listeners, Conditions, and Wait-sets.

- Allows discovery of the presence and QoS info of remote entities and reports this to METT.

- Built-In topics identifying the appearance of corresponding entities:
  - DCPSParticipant
  - DCPSTopic
  - DCPSPublication
  - DCPSSubscription
Limitations

- Focus is on DDS publish-subscribe functionality, limited applicability to large scale fault tolerance study
- Limited to IP transports
- Small network subnet – future efforts may include WAN
- Limited number of hosts currently
  - Dynamic libraries and root permissions require host configuration for runtime loader
  - Multi-use general purpose environment
Evaluation Plan

- Evaluate multiple DDS products as they become available
  - OCI TAO DDS
  - OpenSplice
  - RTI DDS
- Evaluate C++ and Java DDS implementations
- Evaluate DDS MW with various Java Technologies for Real-time
- Evaluate DDS MW supporting additional Operating Systems including RTOSs