Bert Farabaugh
Joe Schlesselman
Real-Time Innovations
Agenda

- Part I. DDS Overview
  - Background
  - Communication model
  - DDS Entities
  - Listeners, Conditions, WaitSets
  - Quality of Service

- Part II. DDS Architecture Update

- Part III. Pervasive Distributed Data
Real-Time Publish-Subscribe DDS

• Efficient mechanism for data communications

Publisher does not need to know who the subscribers are.

Subscribers do not need to know where the data “lives” or continually ask for it.

Producer(s)  Middleware  Consumers
Accessible to all interested applications:

- Data distribution (publishers and subscribers): DDS
- Rich QoS, Automatic discovery and configuration
- Real-time or High-performance
- Peer to Peer Auto Discoverable Network
DDS Standard

- Data Distribution Service for Real-Time Systems
  - Adopted in June 2003
  - Finalized in June 2004
  - Joint submission (RTI, THALES, MITRE, OIS)
  - API specification for Data-Centric Publish-Subscribe communication for distributed real-time systems.
  - Navy OACE, DISR Mandate

- RTI’s role
  - Member of OMG since 2000
  - Co-authors of the original DDS RFP
  - Co-authors of the DDS specification adopted in June 2003
  - Chair of the DDS Finalization Task Force completed March 2004
  - Chair of the DDS Revision Task Force
  - Providers of a COTS implementation of the specification (NDDS.4.0)
Factors driving DDS

- Complex data flows
  - Controlled QoS: rates, reliability, bandwidth
  - Per-node, or per-stream differences
  - Varied transports
  - Direct peer-to-peer transfer
  - Event-driven transfer

- Dynamic configurations
  - Fast location transparency
Factors driving DDS (continued)

**Need for speed**
- Large networks, multicast
- High data rates
- Natural asynchrony
- Tight latency requirements
- Continuously-refreshed data

**Fault tolerance**
- No single-points of failure
- Transparent failover
- Unreliable transports (e.g. wireless)

**Ease of use**
Match the Middleware to the App

- **CORBA**
  - Distributed *object*
    - Client/server
    - Remote method calls
    - Reliable transport
  - Best for
    - Remote command processing
    - File transfer
    - Synchronous transactions

- **DDS**
  - Distributed *data*
    - Publish/subscribe
    - Multicast data
    - Configurable QoS
  - Best for
    - Quick dissemination to many nodes
    - Dynamic nets
    - Flexible delivery requirements

*DDS and CORBA address different needs*
Application Examples
Navy OA projects adopting DDS

- DD(X)
- Ship Self Defense System (SSDS)
- LCS
- Spy OA
- Sea Slice*
- LPD 17*
Future Combat Systems (FCS)

- Has specified DDS as part of SoSCOE
- Large complex real-time data-distribution
- Wireless with intermittent connectivity
- Highly heterogeneous
- Highly dynamic
Data Distribution Service (DDS)

Design Details
Why DDS?

Decoupling
- Location: reduce dependencies
- Redundancy: multiple readers & writers
- Time: data when you want it
- Platform: Connect any set of systems

Benefits
- Modular structure
- Flexibility
- Power
DDS: Publication Subscription Model

Quality of Service: QoS

Domain Participant

Data Reader

Data Writer

Subscriber

Publisher

Data Domain

Node

Topic

Publisher

Subscriber

Data Writer

Data Reader
DDS: Pub/Sub Scenarios

One to One
One to Many
Many to One
Many to Many
## QoS: Quality of Service: Architectural

<table>
<thead>
<tr>
<th>QoS Policy</th>
<th>Concerns</th>
<th>RxO</th>
<th>Changeable</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_DATA</td>
<td>DP,DR,DW</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>TOPIC DATA</td>
<td>T</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>GROUP DATA</td>
<td>P,S</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>ENTITY FACTORY</td>
<td>DP, P, S</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>PRESENTATION</td>
<td>P,S</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>OWNERSHIP</td>
<td>T</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>OWNERSHIP_STRENGTH</td>
<td>DW</td>
<td>N/A</td>
<td>YES</td>
</tr>
<tr>
<td>PARTITION</td>
<td>P,S</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>DURABILITY</td>
<td>T,DR,DW</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>HISTORY</td>
<td>T,DR,DW</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>RESOURCE_LIMITS</td>
<td>T,DR,DW</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
### QoS: Quality of Service: Performance

<table>
<thead>
<tr>
<th>QoS Policy</th>
<th>Concerns</th>
<th>RxO</th>
<th>Changeable</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEADLINE</td>
<td>T,DR,DW</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>LATENCY_BUDGET</td>
<td>T,DR,DW</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>READER DATA LIFECYCLE</td>
<td>DR</td>
<td>N/A</td>
<td>YES</td>
</tr>
<tr>
<td>WRITER DATA LIFECYCLE</td>
<td>DW</td>
<td>N/A</td>
<td>YES</td>
</tr>
<tr>
<td>TRANSPORT PRIORITY</td>
<td>T,DW</td>
<td>N/A</td>
<td>YES</td>
</tr>
<tr>
<td>LIFESPAN</td>
<td>T,DW</td>
<td>N/A</td>
<td>YES</td>
</tr>
<tr>
<td>LIVELINESS</td>
<td>T,DR,DW</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>TIME_BASED_FILTER</td>
<td>DR</td>
<td>N/A</td>
<td>YES</td>
</tr>
<tr>
<td>RELIABILITY</td>
<td>T,DR,DW</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>DESTINATION_ORDER</td>
<td>T,DR</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
Part II. Architecture Update

Evolving Design Patterns
Domain and Domain Participants

• Container for applications that want to communicate

• Applications can join or leave a domain in any order

• New Applications are “Auto-Discovered”

• An application that has joined a domain is also called a “Domain Participant”

Single ‘Domain’ System
**User Application:**
- Creates all DDS entities
- Configures entity QoS
- Associates DW with Topic
- Provides data to DW
DDS Subscription Listener

**User Application:**
- Creates all DDS entities
- Configures entity QoS
- Associates DR with Topic
- Receives Data from DR using a Listener

Diagram:
- **Topic**
- **Listener:** read, take
- **Domain Participant**
- **Data Reader**
- **Subscriber**
  - **Listener**
    - DATA_AVAILABLE
  - **Listener**
    - DATA_ON_READERS
DDS Subscription Wait-Set

**User Application:**
- Creates all DDS entities
- Configures entity QoS
- Associates DR with Topic
- Blocks & waits for data from DR(s) (like select)

**Diagram:**
- Topic
- Domain Participant
- Data Reader
- Subscriber

- Wait for Data
- Read/take

*RTI*
The Filter Expression and Expression Params will determine which instances of the Topic will be received by the subscriber.
Part III. Pervasive Distributed Data

Embedded to Enterprise Bridging Database Interoperability
A Standards Based Global Data Space

- Data Accessible to all interested applications:
  - Data distribution (publishers and subscribers): DDS
  - Data management (storage, retrieval, queries): SQL
  - Rich QoS, Automatic discovery and configuration
  - Real-time and/or High-performance access to data
Embedded to Enterprise Bridging

- RT SQL App
- SkyBoard
- DBMS
- NDDS
- Network
- NDDS
- RT DDS App
- SQL App
- SkyBoard
- DBMS
- NDDS
- Enterprise App
- DBMS
- DSMS
- Web Service
- Enterprise Application
- NDDS
- Embedded Edge Device
**Summary**

**DDS targets applications that need to distribute data in a real-time environment**

**DDS is highly configurable by QoS settings**

**DDS provides a shared “global data space”**
- Any application can publish data it has
- Any application can subscribe to data it needs
- Automatic discovery
- Facilities for fault tolerance
- Heterogeneous systems easily accommodated

**Pervasive Data Distributed**
- Embedded to Enterprise Bridging
- Database Interoperability
Thank you for your time today!

bert.farabaugh@rti.com
joe.schlesselman@rti.com