Intelligent Event Processing in Quality of Service (QoS) Enabled Publish/Subscribe (Pub/Sub) Middleware

Joe Hoffert
jhoffert@dre.vanderbilt.edu
http://www.dre.vanderbilt.edu/~jhoffert/

CS PhD Student
Vanderbilt University
Nashville, Tennessee
Intelligent Event Processing

Context: Event Processing Systems
- Wide range of application domains
  - Ambient assisted living
  - Fractionated satellite systems
  - Weather monitoring
  - Disaster recovery
  - Stock quote update systems ...

Challenge: Supporting QoS & Managing Complexity
- Wide range of QoS needed
- Examples: low latency, reliable event delivery, coordinated data streams, fault tolerance
- Interacting QoS demands

Solution: QoS-enabled Pub/Sub Middleware
- Intelligent processing of events
  - QoS primitives for low level functionality
  - Patterns for higher level functionality
QoS-enabled Pub/Sub Middleware Case Study

Object Management Group’s Data Distribution Service (DDS)

DDS provides flexibility, power, & modular structure by decoupling:

- **Location** – anonymous pub/sub
- **Redundancy** – any number of readers & writers
- **QoS** – async, disconnected, time-sensitive, scalable, & reliable data distribution at multiple layers
- **Platform & protocols** – portable & interoperable
DDS Domains & Domain Participants

- The domain is the basic construct used to bind individual applications together for communication
  —Like a VPN

![Diagram with nodes and domains]
DDS Entities include

- **DomainParticipants**
  - Entry points
- **Topics**
  - Typed data
- **Publishers**
  - Manage **DataWriters**
- **Subscribers**
  - Manage **DataReaders**

- Data can be accessed in two ways
  - Wait-based (synchronous calls)
  - Listener-based (asynchronous callbacks)
- Sophisticated support for filtering
  - e.g., **Topic**, **Content-FilteredTopic**, or **MultiTopic**
- Configurable via (many) QoS policies
QoS Policies Supported by DDS

- DDS entities (i.e., topics, data readers/writers) configurable via QoS policies
- QoS tailored to data distribution in distributed realtime & embedded (DRE) information systems
  - DEADLINE
    - Establishes contract regarding rate at which periodic data is refreshed
  - LATENCY_BUDGET
    - Establishes guidelines for acceptable end-to-end delays
  - TIME_BASED_FILTER
    - Mediates exchanges between slow consumers & fast producers
  - RESOURCE_LIMITS
    - Controls resources utilized by service
  - RELIABILITY (BEST_EFFORT, RELIABLE)
    - Enables use of real-time transports for data
  - HISTORY (KEEP_LAST, KEEP_ALL)
    - Controls which (of multiple) data values are delivered
  - DURABILITY (VOLATILE, TRANSIENT, PERSISTENT)
    - Determines if data outlives time when they are written
    - ... and many more ...

- Request/offered compatibility checked by DDS, helps to manage complexity
Types & Keys

- A DDS **Type** is represented by a collection of data items.
  - e.g. “IDL struct” in the CORBA Platform Specific Model (PSM)
    ```
    struct AnalogSensor {
        string sensor_id; // key
        float value; // other sensor data
    };
    ```
- A subset of the collection is designated as the **Key**.
  - The Key can be indicated by IDL annotation in CORBA PSM, e.g.,
    ```
    #pragma DDS_KEY AnalogSensor::sensor_id
    ```
- The type is manipulated by means of automatically-generated typed interfaces.
  - IDL compiler may be used in CORBA PSM implementation
- A Type is associated with generated code:
  - AnalogSensorDataWriter // write values
  - AnalogSensorDataReader // read values
  - AnalogSensorType // can register itself with Domain
Topics

- A DDS Topic is the connection between data writers & data readers.

- A Topic is comprised of a Name and a Type.
  - Name must be unique in the Domain.
  - Many Topics can have the same Type.

- Provision is made for content-based subscriptions.
  - MultiTopics correspond to SQL join
  - Content-Filtered Topics correspond to SQL select.
Data Writers & Publishers

- Data Writers are the primary access point for an application to publish data into a DDS data domain
- The Publisher entity is a container to manage one or more Data Writers
- Publishers & Data Writers can have their own QoS policies
- User applications
  - Associate Data Writers with Topics
  - Provide data to Data Writers
  - Data is typically defined using OMG IDL
    - As strongly or weakly typed as desired

---

Fig. 7 Publication Model
Data Readers & Subscribers

• A Data Reader is the primary access point for an application to access data that has been received by a Subscriber

• Subscriber is used to manage one or more Data Readers

• Subscribers & Data Readers can have their own QoS policies

• User applications
  – Associate Data Readers with Topics
  – Receive data from Data Readers using Listeners (async) or Wait-Sets (sync)
Overview of the Data Distribution Service (DDS)

- DDS is an highly efficient OMG pub/sub standard
  - e.g., fewer layers, less overhead
Overview of the Data Distribution Service (DDS)

• DDS is an highly efficient OMG pub/sub standard
  • e.g., fewer layers, less overhead
• DDS provides meta-events for detecting dynamic changes
Overview of the Data Distribution Service (DDS)

- DDS is an highly efficient OMG pub/sub standard
  - e.g., fewer layers, less overhead
- DDS provides meta-events for detecting dynamic changes
- DDS provides policies for specifying many QoS requirements of tactical information management systems, e.g.,
  - Establish contracts that precisely specify a wide variety of QoS policies at multiple system layers
Overview of the Data Distribution Service (DDS)

• DDS is an highly efficient OMG pub/sub standard
  • e.g., fewer layers, less overhead
• DDS provides meta-events for detecting dynamic changes
• DDS provides policies for specifying many QoS requirements of tactical information management systems, e.g.,
  • Establish contracts that precisely specify a wide variety of QoS policies at multiple system layers
  • Move processing closer to data
All QoS Policies in DDS

- Deadline
- Destination Order
- Durability
- Durability Service
- Entity Factory
- Group Data
- History
- Latency Budget
- Lifespan
- Liveliness
- Ownership
- Ownership Strength
- Partition
- Presentation
- Reader Data Lifecycle
- Reliability
- Resource Limits
- Time-Based Filter
- Topic Data
- Transport Priority
- User Data
- Writer Data Lifecycle

Detailed explanations in DDS specification
Best Effort Reliability QoS in DDS

QoS Reliability = BEST_EFFORT

- Very predictable
- Data is sent without reply
- Publishers and subscribers match and obey QoS Deadline policy settings
- Time-based Filter QoS policy gives bandwidth control

Notification of new data objects

‘Global’ Data Store

Publisher

Subscriber

Subscriber

Subscriber

notification

time-based filter

no notification

notification

deadline

timeout

time
Reliable QoS in DDS

QoS Reliability = RELIABLE

Ordered instance delivery is guaranteed
• **DataWriter** is bound (at creation time) to a single **Topic**

• **DataReader** is bound (at creation time) with one or more topics (**Topic**, **ContentFilteredTopic**, or **MultiTopic**)
  
  – **ContentFilteredTopic** & **MultiTopic** provide means for content-based subscriptions & “joins”, respectively
Content-based Subscriptions

• **ContentFilteredTopic** (like a DB-selection)
  – Enables subscriber to only receive data-updates whose value verifies a condition.
  – e.g. subscribe to “Pressure” of type `AnalogData`
    – where “value > 200”

• **MultiTopic** (like a DB-join operation)
  – Enables subscription to multiple topics & re-arrangement of the data-format
  – e.g. combine subscription to “Pressure” & “Temperature” & re-arrange the data into a new type:
    ```
    struct { float pres; float temp; }
    ```
DDS Content-Filtered Topics

Filter Expression and Expression Params determine which Topic instances the Subscriber receives.

Filter Expression:
Value > 260

Expression Params

<table>
<thead>
<tr>
<th>Instance</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance 1</td>
<td>249</td>
</tr>
<tr>
<td>Instance 2</td>
<td>230</td>
</tr>
<tr>
<td>Instance 3</td>
<td>275</td>
</tr>
<tr>
<td>Instance 4</td>
<td>262</td>
</tr>
<tr>
<td>Instance 5</td>
<td>258</td>
</tr>
<tr>
<td>Instance 6</td>
<td>261</td>
</tr>
<tr>
<td>Instance 7</td>
<td>259</td>
</tr>
</tbody>
</table>
DDS MultiTopic Subscriptions

MultiTopics can combine, filter, and rearrange data from multiple Topics.
QoS Pattern for Consistency

Goal: Have all data readers receive same data in same order from all data writers even in event of crash

Approach: Compose support for consistency using QoS primitives

DURABILITY
— Set to PERSISTENT
— Data survives crash of the data writers/readers

RELIABILITY
— Set to RELIABLE
— Data sent from single writer received in order sent

DESTINATION_ORDER
— Set to SOURCE_TIMESTAMP
— Data sent from multiple writers received in order sent

LIFESPAN
— Set to INFINITE
— Designate data as valid until taken by reader

Consistency Pattern

- Durability
- Reliability
- Destination Order
- Lifespan
QoS Pattern for Fault Tolerance

Goal: Have data writer fail over to replica if non-responsive

Approach: Compose support for fault tolerance using QoS primitives

**LIVELINESS**
- Set to AUTOMATIC
- Loss of heartbeat triggers failover

**DEADLINE**
- PERIOD set to maximum latency of data
- Data not received within period triggers failover

**OWNERSHIP**
- Set to EXCLUSIVE
- Data only received from one data writer

**OWNERSHIP_STRENGTH**
- VALUE reflects order/importance of replica
- Determines who takes over after fault
Concluding Remarks

Supporting & managing intelligent event processing can be challenging

- Wide range of QoS needed
- Complex interactions

QoS-enabled Pub/Sub middleware can help

- QoS primitives for low level functionality
- Patterns for higher level functionality

Additional resources

- OMG DDS specification (portals.omg.org/dds)
- DDS patterns
  - [www.prismtechnologies.com/section-item.asp?snun=5&sid=83](http://www.prismtechnologies.com/section-item.asp?snun=5&sid=83)

Thank you for your time and attention

Soli Deo Gloria