DDS Use Cases: Effective Application of DDS Patterns and QoS

OMG Workshop: 10-13 July, 2006

Gordon A. Hunt
Principal Engineer
gordon.hunt@rti.com
www.rti.com
Outline

- Quality of Service
  - What make DDS Different?
  - Quality of Service (QoS) Review

- Patterns
  - Continuous Data
  - State Data
  - Alarm/Event Data
  - Hot-swap and failover
  - Controlled Data access
  - Filtered by Data content
What Makes DDS Different?

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

Benefits
- Modular structure
- Flexibility
- Power
QoS: Quality of Service

• Definition:
  - Quality of Service (QoS) policies permit application to manage, prioritize and shape data-flow in a network.

• Function:
  - Standard DDS QoS parameters address general aspects of data flow:
    - Data Volatility
    - Data Delivery
    - Fault-tolerance
    - Discovery
    - System Resources (Resource limits, Transport configuration)
  - Vendor specific extended QoS provide access to underlying implementation.
# QoS: Quality of Service

<table>
<thead>
<tr>
<th>QoS Policy</th>
<th>QoS Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DURABILITY</td>
<td>USER DATA</td>
</tr>
<tr>
<td>HISTORY</td>
<td>TOPIC DATA</td>
</tr>
<tr>
<td>READER DATA LIFECYCLE</td>
<td>GROUP DATA</td>
</tr>
<tr>
<td>WRITER DATA LIFECYCLE</td>
<td>PARTITION</td>
</tr>
<tr>
<td>LIFESPAN</td>
<td>PRESENTATION</td>
</tr>
<tr>
<td>ENTITY FACTORY</td>
<td>DESTINATION ORDER</td>
</tr>
<tr>
<td>RESOURCE LIMITS</td>
<td>OWNERSHIP</td>
</tr>
<tr>
<td>RELIABILITY</td>
<td>OWNERSHIP STRENGTH</td>
</tr>
<tr>
<td>TIME BASED FILTER</td>
<td>LIVELINESS</td>
</tr>
<tr>
<td>DEADLINE</td>
<td>LATENCY BUDGET</td>
</tr>
<tr>
<td>CONTENT FILTERS</td>
<td>TRANSPORT PRIORITY</td>
</tr>
</tbody>
</table>
QoS: Quality of Service

- DDS permits QoS to be associated with each DDSEntity
  - Topic (T)
  - Data Reader (DR)
  - Data Writer (DW)
  - etc.

- DDS QoS semantics
  - DataWriter offers available level of QoS
  - DataReader requests needed level of QoS
  - offered level ≥ requested level
QoS Contract “Request / Offered”

Ensure that the compatible QoS parameters are set.

Global Data Space
Using QoS to Send Data

- **Continuous Data**
  - Constantly updating data
  - Many-to-many delivery
  - Sensor data, last value is best
  - Seamless failover

- **State Information**
  - Occasionally changing persistent data
  - Recipients need latest and greatest

- **Alarms & Events**
  - Asynchronous messages
  - Need confirmation of delivery
Using QoS to Send Data

- **Continuous Data**
  - Constantly updating data – best-effort
  - Many-to-many delivery – keys, multicast
  - Sensor data, last value is best – keep-last
  - Seamless failover – ownership, deadline

- **State Information**
  - Occasionally changing persistent data – durability
  - Recipients need latest and greatest – history

- **Alarms & Events**
  - Asynchronous messages – liveliness
  - Need confirmation of delivery – reliability
Design Patterns – Continuous Data

- **Scenarios**
  - Multiple Writers constantly updating the topic “Data”
  - Many-to-many delivery
  - Desired to have separate instances of “Data”

- **Example**
  - Track updates, vehicle position, stock quotes, temperatures,…

**Quality of Service**

- **Best Effort**
  - Delivery mechanism can be specified on a per-Reader basis

- **Keys**
  - Controls scalability, enabling single topic with multiple instances

- **Multicast**
  - Leverage efficiencies in transports on a per-Reader basis
Best Effort Communications

QoS: Reliability = Best Effort

- Low latency and overhead
- Samples are sent without any feedback
- Deadline specifies determinism
- Time-Based Filter controls bandwidth
Keys (data-object scalability)

- Multiple instances of the same topic

- Do not need a separate Topic for each data-object instance
- Used to sort specific instances

- Topic key can be any data-type within the Topic.

Example:

long ID //@key
long pos_x
long pos_y
Design Patterns – Continuous Data

- Scenarios (Part 2)
  - A DataReader need to know when a writer or instance ‘fails’
  - Some DataReaders need to limit their updates
  - Only one DataWriter can update a specific instance

- Examples
  - Two radars monitoring the same Track, multiple sources of weather data, manual override of UAV flight control,…
  - High level display doesn’t need all real-time data

Quality of Service

- Deadline
  - Can indicate health and send/receive time requirements

- Time Based Filter
  - Readers can control their own data delivery rate

- Ownership
  - Controls exclusivity of writing to an instance
QoS: Deadline

Specified as a “period”, following request/offered design pattern

- **Publisher** (Data Writer)
  - Commits to provide data each deadline period.

- **Subscriber** (Data Reader)
  - Expects data every deadline period.

- **Listener**
  - Failed to get data

- **Domain Participant**

(c) 2006 Real-Time Innovations, Inc.
QoS: Time Based Filter

“minimum_separation”: DataReader does not want to receive data faster than the min_separation time.
QoS: Ownership

 Specifies whether more than one DataWriter can update the same instance of a data-object

**Ownership = EXCLUSIVE**
“Only highest-strength data writer can update each data-instance”

**Ownership = SHARED**
“All data-writers can each update data-instance”

Provides fast, robust, transparent replacement for fail-over and/or take-over.
Design Patterns – State Information

- Scenarios
  - One DataWriter of “Data”, multiple DataReaders
  - Late joining DataReaders should get last set of “Data”
  - Every DataReader should get all issued data

- Examples
  - Arrival and departure time for trains, current aircraft flight plans, operational system configuration and mode

Quality of Service
- History (Keep Last)
  - Controls how much data need to be available
- Reliability
  - Ensures that data is delivered
- Durability & Lifespan
  - Controls persistence, how and where data is stored
QoS: History – Last x or All

**KEEP_ALL:**
Publisher: keep all until delivered
Subscriber: keep each sample until the application processes that instance

**KEEP_LAST:** “depth” integer for the number of samples to keep at any one time

![Diagram showing QoS history options: Keep All, Keep Last 2, Keep Last 4.](c) 2006 Real-Time Innovations, Inc.
Reliable Communications

Reliability QoS: Ordered Instance delivery is guaranteed

Good for: Command and Event Data

Global Data Space

(c) 2006 Real-Time Innovations, Inc.
Reliable Communications

Reliability QoS: Ordered Instance delivery is guaranteed

Global Data Space

(c) 2006 Real-Time Innovations, Inc.
QoS: Durability

Durability Kind:
- VOLATILE – No Instance History Saved
- TRANSIENT – History Saved in Local Memory
- PERSISTENT – History Saved in Permanent storage

Durability determines if/how instances of a topic are saved.

# saved in Transient affected by QoS: History and QoS: Resource_Limits
QoS: Lifespan

duration
Manages samples in the history queues, attached to each Sample
Design Patterns – Alarms & Events

- **Scenarios**
  - All issues of data need to be received
  - Asynchronous messages
  - Possible only one authorized ‘Event’ Writer

- **Examples**
  - Emergency notifications, boiler over heating alarm, calling the elevator, smoke detectors,…

*Quality of Service*

- **Liveliness**
  - Need to know if a DataWriter is present independent of receipt of data

- **Reliability, History (Keep All)**
- **Ownership**
- **Lifespan**
QoS: Liveliness – Type and Duration

Type: Controls who is responsible for issues of ‘liveliness packets’
AUTOMATIC = Infrastructure Managed
MANUAL = Application Managed

- Failed to renew lease
- lease_duration
- Liveliness Message
- LP
- S
- Topic
Design Patterns – Hot-swap and failover

- Scenarios
  - Primary / Secondary are duplicate machines
    - Running duplicate applications
    - Primary handles full load until failure
  - Need to selectively override Topics

- Examples
  - Know that train operator is still awake, Emergency notifications, boiler over heating alarm, calling the elevator, smoke detectors,…

Quality of Service

- Liveliness
  - Need to know if a DataWriter is present independent of receipt of data

- Deadline
  - Need to refresh every instance continuously

- Ownership
  - Switch primary/secondary data sources
Design Patterns – Controlled Data Access

- **Scenarios**
  - Limited access to certain data by node or application or entity or user ID,…
  - Exchange meta-data for authentication and identification
  - Execution of identically configured systems

- **Examples**
  - Command authorization, configuration control, Separation of simulation and operational data
  - Separation of simulation and operational data
  - Isolation of data from multiple system instances (e.g. each tank will manage and see its own data, despite using the same Topics)

**Quality of Service**

- **Domains**
  - A separate global data space for each Domain

- **Partitions**
  - Dynamic association of Readers and Writers (grouping)

- **User Data**
  - Exchange Reader and Writer meta-data and with Discovery

- **Ignore API**
  - Permanent denial of read/write access
Domain and Domain Participants

Using Multiple domains for Scalability, Modularity & Isolation

Multiple Domain System
QoS: Partition

Partitions are a logical “namespace” for topics
** Partition string names must match between publisher and subscriber
QoS: User Data

- USER_DATA is contained within the DDS metadata.
- User data could be used to authenticate an origination entity or pass additional attribute information.

Entity:
- Domain Participant (User_Data)
- DataReader (Group_Data)
- DataWriter (Topic_Data)

Process declarations based on user data.
Use `ignore_xxx` API.
Design Patterns – Filter by Data Content

- **Scenarios**
  - Receive only ‘interesting’ Data
  - Send a message to a specific receiver(s)

- **Examples**
  - Monitor aircraft within region of interest, display only hostile tracks, monitor objects near my current ship location,…
  - Response to authorization gas pump sale request

**Quality of Service/API**

- **Content Filtered Topics**
  - SQL filter language with changeable filter parameters
  - Encode receiverID in message send

- **read/take() with a QueryCondition**
  - By specific instance or lifecycle states
Content Filtered Topics

The Filter Expression and Expression Params will determine which instances of the Topic will be received by the subscriber.
Thank You!

Questions?

Gordon A. Hunt
Principal Engineer
gordon.hunt@rti.com
www.rti.com