DDS on SPARTA

Stefano Zampieri

szampier@eso.org
Agenda

- DDS Overview
- DDS on SPARTA
- Network considerations
- Conclusions
DDS in a nutshell

- DDS is a Real-Time Data-Centric Networking Middleware
- DDS focuses on
  - Performance
    - High-performance data-access APIs (zero copy access)
  - Configurability
    - Quality of Service
  - Scalability
    - UDP, multicast, reliable multicast
  - DDS does not require the presence of intermediate brokers
    - Applications can communicate directly peer-to-peer
  - DDS supports advanced features
    - E.g. source filtering (via Content-based and Time-based filters)
  - Integration
    - E.g. with Database Management Systems
DDS Standards

- Data Distribution Service for Real-Time Systems (DDS)
  - API specification for Data-Centric Publish-Subscribe communication for distributed real-time systems.
  - Current version 1.2

- DDS Interoperability wire Protocol (DDSI/RTPS)
  - Ensure that applications based on different vendors’ implementations of DDS can interoperate.
  - Current version 2.1

- Related Standards
  - UML Profile for DDS adopted June 2008
  - DDS for light weight CCM adopted 2008
  - Extensible and Dynamic Topic Types for DDS adopted 2010

- Standards under Development
  - Native Language C++ API for DDS
  - DDS-Java
DDS Vendors

- Real-Time innovations, Inc. (Commercial, Open Community Source)
- PrismTech (Commercial & Open Source)
- Object Computing, Inc. (OpenDDS, Open Source)
- Twin Oaks Computing, Inc. (CoreDX, Commercial)
- Etc.
DDS Model

- Topic
  - Data Writer
    - Publisher
    - Offered QoS
      - Domain Participant
  - Listeners
- Topic
  - Data Reader
    - Subscriber
    - Requested QoS
      - Domain Participant
  - Listeners

Global Data Space
## 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>

**Real Time Control Workshop, 4-5 Dec 2012**
Example QoS

```xml
<durability>
  <kind>DDS_TRANSIENT_LOCAL_DURABILITY_QOS</kind>
</durability>

<time_based_filter>
  <minimum_separation>
    <sec>1</sec>
    <nanosec>0</nanosec>
  </minimum_separation>
</time_based_filter>

<history>
  <kind>DDS_KEEP_ALL_HISTORY_QOS</kind>
</history>

<reliability>
  <kind>DDS_RELIABLE_RELIABILITY_QOS</kind>
  <max_blocking_time>
    <sec>0</sec>
    <nanosec>0</nanosec>
  </max_blocking_time>
</reliability>
```
Agenda

- DDS Overview
- DDS on SPARTA
- Network considerations
- Conclusions
SPARTA Real Time Data Flows

Recorder

Display

Aux. Task

Calibration

Recorder

Display

DDS VisPixelFrame

DDS VisLoopFrame

DDS Disturbance Frame

VisPixelConcentr

VisLoopConcentr

Uploader

WPU

WPU

RECN

CTR

DM commands

Real Time Box
Low latency

WFS pixels

Cluster

Real Time Control Workshop, 4-5 Dec 2012
### Additional Data Flows

- **DB Events (n:m)**
  - Database updates sent to DBGateway and to Main
  - Including Alarms
  - ~300 events/s (measured on SPHERE)

- **CDMS Events (1:n)**
  - Upon object updates in the SPARTA Cfg. DB
  - Trigger chain of events

- **Log Events (n:1)**
  - Log messages sent to LogGateway
Throughput requirements

- **SAXO**
  - VisLoop: 20KB @ 1.2KHz
  - VisPixel: 112KB @ 10Hz
  - Tot: ~25MB/s

- **AOF**
  - LGSSLoop: 67KB @ 1KHz
  - LGSPixel: 450KB @ 10 Hz
  - Tot: ~72MB/s

- **Multicast !**
SPARTA DDS Model

- **SPARTA DDS Wrapper** (spadds)
  - Simplified API (DDS-like)
  - **Publisher** *(write)*
  - **Subscriber** + **DataListener** *(onDataAvailable)*
  - **Topic**: template parameter + string
  - QoS defined in XML configuration file
  - QoS Profiles, referenced by name when creating Publishers and Subscribers
    - *HighThroughputReliableProfile* (reliable, large send queue)
    - *LargePacketsReliableProfile* (>64KB, asynchronous publisher)
    - *ReliableEventProfile* (durability)
    - *PixelDisplayProfile* (time based filter)
SPARTA Data Task

- Simplifies development of data tasks
- Simple model: receive N samples then process them in a separate thread
- Developer must implement virtual methods `received_`, `process_`, and `deadlineMissed_`
- Examples: Garbage Collector, Loop Optimiser, Atmospheric Monitor, etc
Agenda

- DDS Overview
- DDS on SPARTA
- Network considerations
- Conclusions
Multicast & IGMP

Multicast vs Broadcast

IP Multicast
- Take advantage of multicast efficiency in network
- IP address range: 224.0.0.0 to 239.255.255.255.

IGMP snooping switches
- No IGMP snooping
  - Multicast traffic broadcasted to each port
- IGM Snooping
  - Switch forwards multicast packets to correct ports
    - Monitors IGMP join messages
  - Multicast addresses configured by subscriber
### Wireshark & RTPS2

#### Filter:
```
No.  Time          Source          Destination          Protocol      Length      Info
```

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>2</td>
<td>0.000000</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>3</td>
<td>0.012035</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>4</td>
<td>0.012051</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>5</td>
<td>0.023999</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>6</td>
<td>0.024013</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>7</td>
<td>0.024123</td>
<td>192.168.10.10</td>
<td>192.168.10.10</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>8</td>
<td>0.024123</td>
<td>192.168.10.10</td>
<td>192.168.10.10</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>9</td>
<td>0.035956</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>10</td>
<td>0.035968</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>11</td>
<td>0.047972</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>12</td>
<td>0.047978</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
<tr>
<td>13</td>
<td>0.060033</td>
<td>192.168.10.10</td>
<td>192.168.10.30</td>
<td>RTPS2</td>
<td>110</td>
<td>INFO_DST, HEARTBEAT</td>
</tr>
</tbody>
</table>

Frame 1: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
User Datagram Protocol, Src Port: 46468 (46468), Dst Port: 10417 (10417)
Real-Time Publish-Subscribe Wire Protocol 2.x
Scaling up

- Initial tests on 10 GigE using rtiperf (no tuning)

<table>
<thead>
<tr>
<th></th>
<th>One way Latency (us)</th>
<th>Packets/s</th>
<th>Mb/s</th>
<th>Packet loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best effort Small packets</td>
<td>105</td>
<td>130000</td>
<td>104</td>
<td>Very low</td>
</tr>
<tr>
<td>Reliable Small packets</td>
<td>510?</td>
<td>35000</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>Best effort Large packets</td>
<td>357</td>
<td>11160</td>
<td>5620</td>
<td>About 1%</td>
</tr>
<tr>
<td>Reliable Large packets</td>
<td>372</td>
<td>10400</td>
<td>5000</td>
<td>0</td>
</tr>
</tbody>
</table>

- Jumbo frames?
Agenda

- DDS Overview
- DDS on SPARTA
- Network stuff
- Conclusions
Conclusions

- DDS works (and saves development)
  - Only 1 serious issue up to now, solved by upgrading
  - Reliable (no intermediate brokers), efficient
  - Simple programming model, also thank to wrapper API
  - Highly configurable, through external QoS

- Future perspectives
  - DDS/RTPS on Real-Time Box?
Questions

Thank You!

References