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

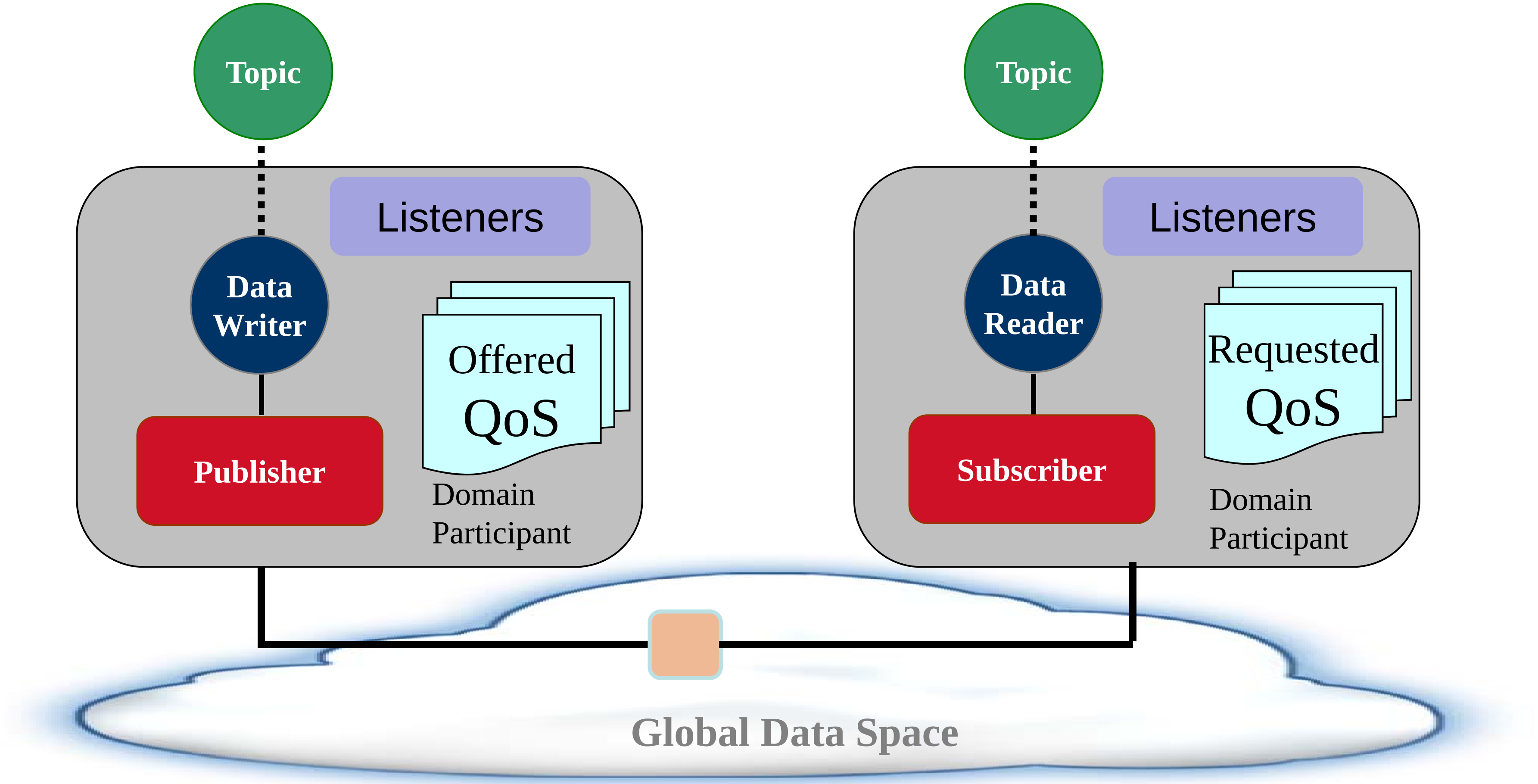
DDS Standards

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

QoS: Quality of Service

	QoS Policy	QoS Policy	
	DURABILITY	USER DATA	User QoS
Volatility	HISTORY	TOPIC DATA	
	READER DATA LIFECYCLE	GROUP DATA	
	WRITER DATA LIFECYCLE	PARTITION	Presentation
	LIFESPAN	PRESENTATION	
Infrastructure	ENTITY FACTORY	DESTINATION ORDER	
	RESOURCE LIMITS	OWNERSHIP	Redundancy
	RELIABILITY	OWNERSHIP STRENGTH	
Delivery	TIME BASED FILTER	LIVELINESS	Transport
	DEADLINE	LATENCY BUDGET	
	CONTENT FILTERS	TRANSPORT PRIORITY	

Example QoS

```
<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>
```

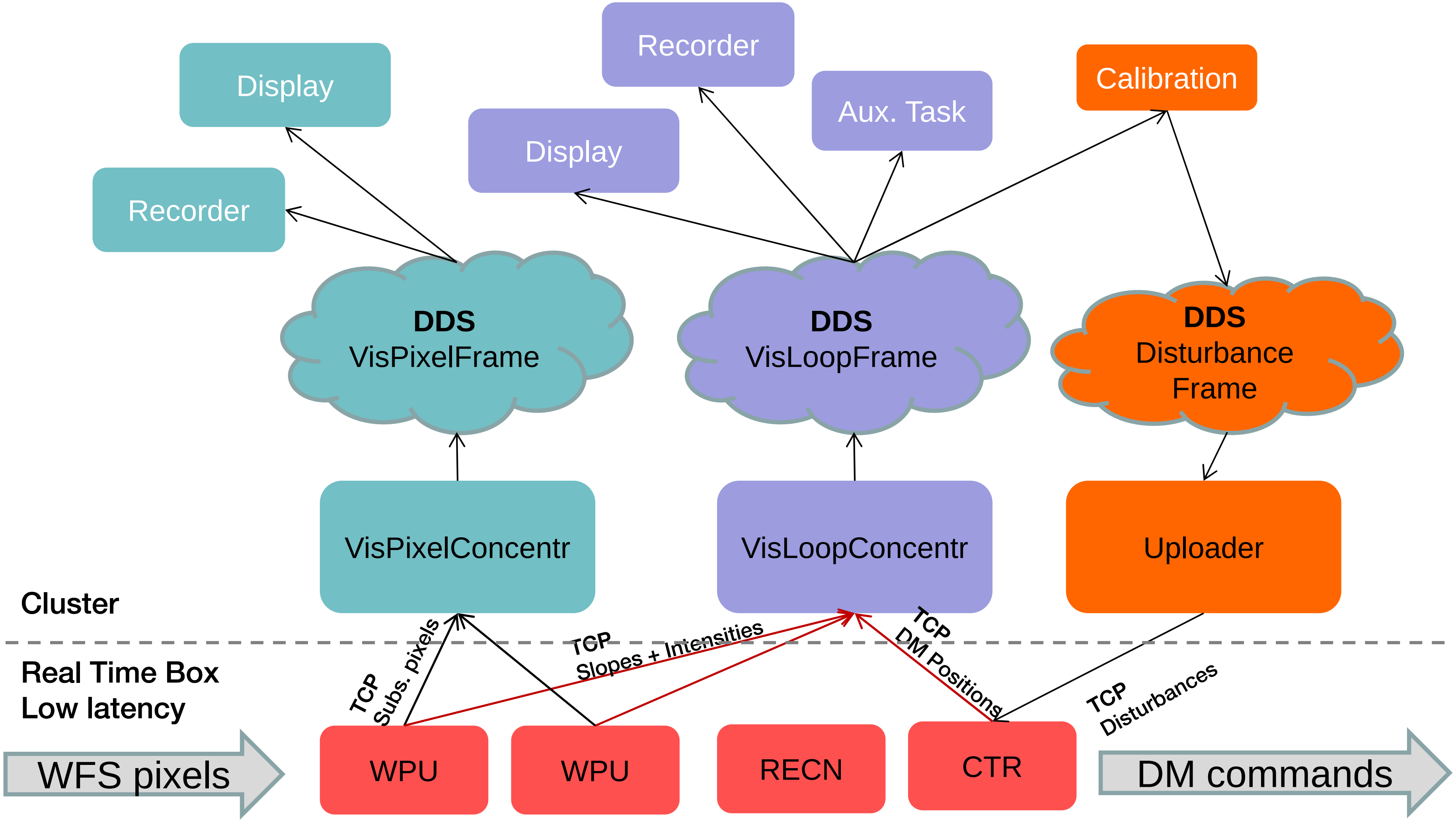
```
<history>
        <kind>DDS_KEEP_ALL_HISTORY_QOS</kind>
</history>
```

```
<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



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

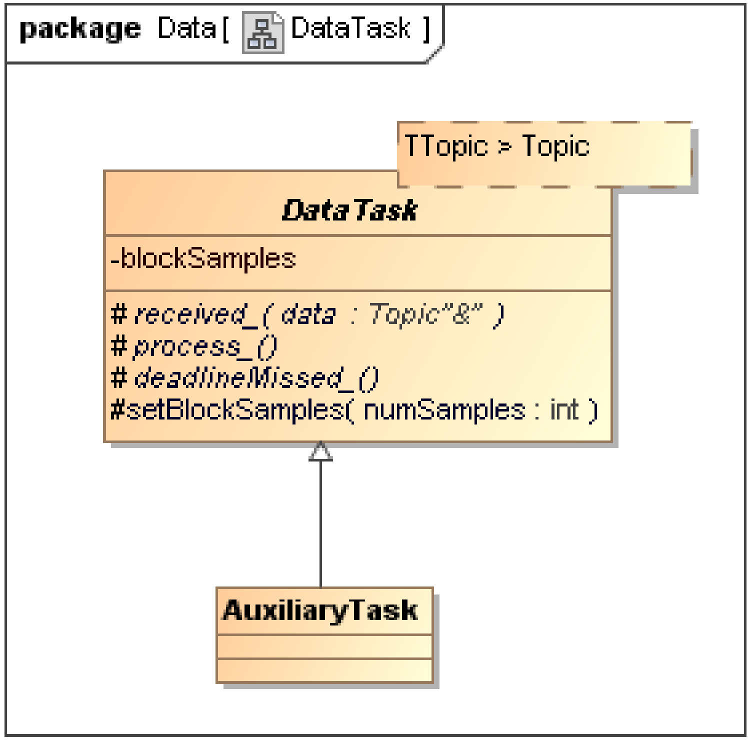
- Multicast !

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

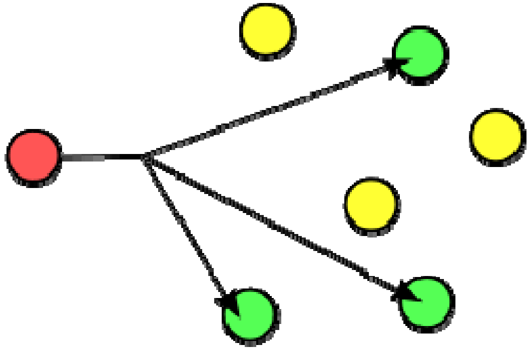
- Simplifies development of data tasks
- Simple model: receive N samples then process them in a separate thread
- Developer must implement virtual methods *received_*, *process_*, and *deadlineMissed_*
- Examples: Garbage Collector, Loop Optimiser, Atmospheric Monitor, etc


package Data [DataTask]
TTopic > Topic
DataTask
-blockSamples
# received_(data : Topic"&")
# process_()
# deadlineMissed_()
#setBlockSamples(numSamples : int)
AuxiliaryTask

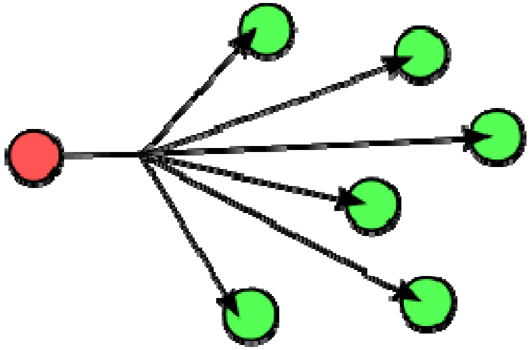
Agenda

- DDS Overview
- DDS on SPARTA
- Network considerations
- Conclusions

Multicast & IGMP



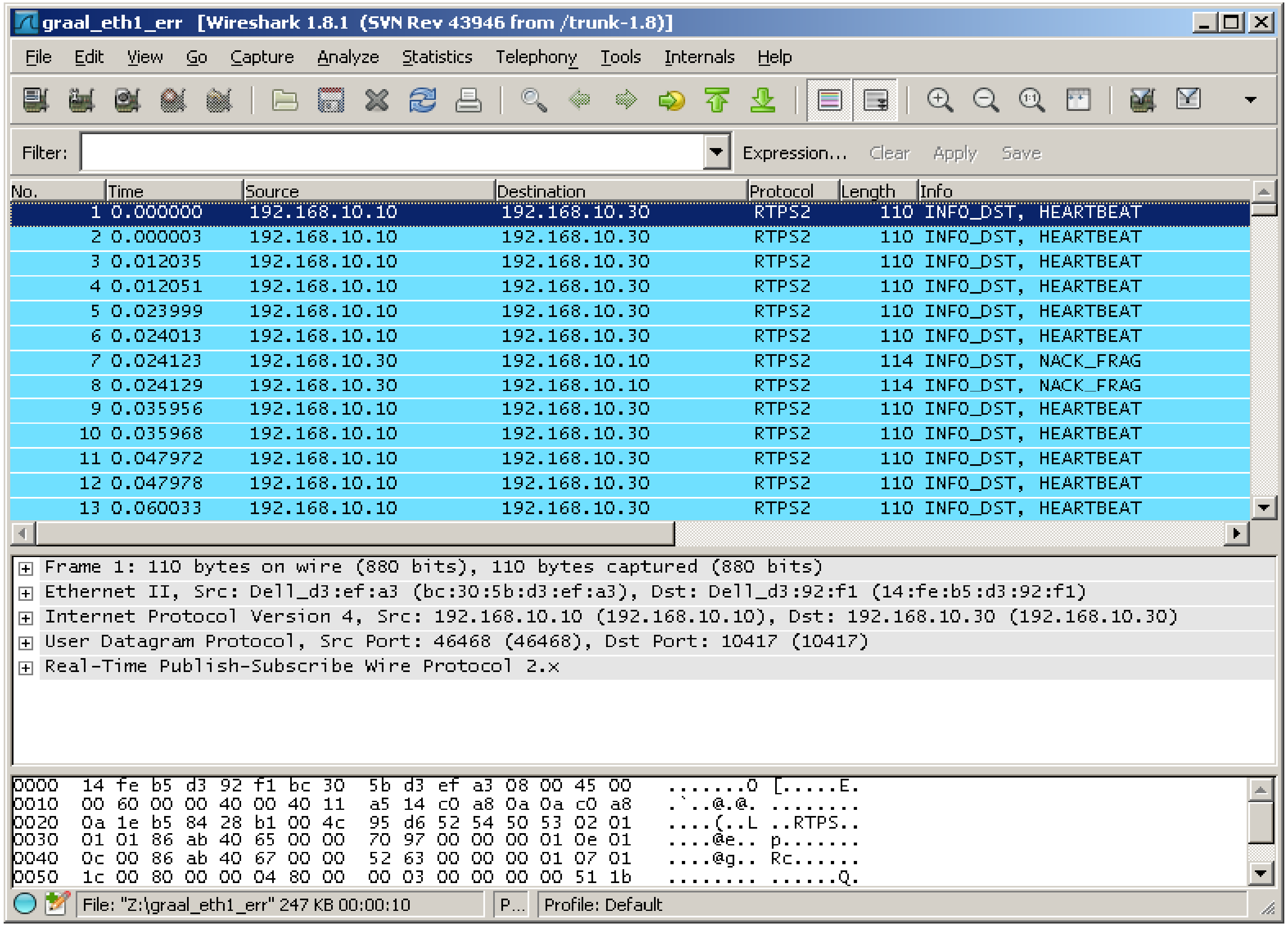
Multicast vs Broadcast



- 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


graal_eth1_err [Wireshark 1.8.1 (SVN Rev 43946 from /trunk-1.8)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.     Time        Source          Destination     Protocol  Length Info
      1 0.000000    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
      2 0.000003    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
      3 0.012035    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
      4 0.012051    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
      5 0.023999    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
      6 0.024013    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
      7 0.024123    192.168.10.30   192.168.10.10   RTPS2        114 INFO_DST, NACK_FRAG
      8 0.024129    192.168.10.30   192.168.10.10   RTPS2        114 INFO_DST, NACK_FRAG
      9 0.035956    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
     10 0.035968    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
     11 0.047972    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
     12 0.047978    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
     13 0.060033    192.168.10.10   192.168.10.30   RTPS2        110 INFO_DST, HEARTBEAT
```

```
⊞ Frame 1: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
⊞ Ethernet II, Src: Dell_d3:ef:a3 (bc:30:5b:d3:ef:a3), Dst: Dell_d3:92:f1 (14:fe:b5:d3:92:f1)
⊞ Internet Protocol Version 4, Src: 192.168.10.10 (192.168.10.10), Dst: 192.168.10.30 (192.168.10.30)
⊞ User Datagram Protocol, Src Port: 46468 (46468), Dst Port: 10417 (10417)
⊞ Real-Time Publish-Subscribe Wire Protocol 2.x
```

```
0000  14 fe b5 d3 92 f1 bc 30  5b d3 ef a3 08 00 45 00   .......0 [.....E.
0010  00 60 00 00 40 00 40 11  a5 14 c0 a8 0a 0a c0 a8   .`..@.@. ........
0020  0a 1e b5 84 28 b1 00 4c  95 d6 52 54 50 53 02 01   ....(..L ..RTPS..
0030  01 01 86 ab 40 65 00 00  70 97 00 00 00 01 0e 01   ....@e.. p.......
0040  0c 00 86 ab 40 67 00 00  52 63 00 00 00 01 07 01   ....@g.. Rc......
0050  1c 00 80 00 00 04 80 00  00 03 00 00 00 00 51 1b   ........ ......Q.
```

File: "Z:\graal_eth1_err" 247 KB 00:00:10 | P... | Profile: Default

Scaling up

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

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

- 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

http://portals.omg.org/dds/sites/default/files/DDS_Tutorial_RT_Worskshop_2010.pdf