Data Distribution Service as an alternative to CORBA Notification Service for the Alma Common Software

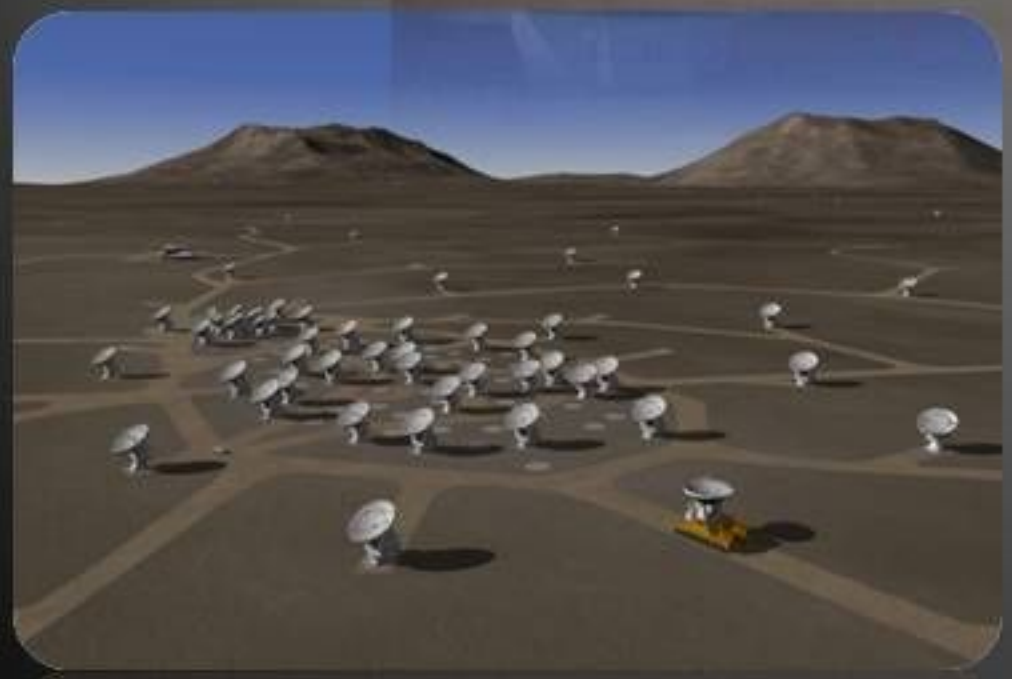
Jorge A. Avarias Alfaro
(ALMA UTFSM group/NRAO)
javarias@inf.utfsm.cl
Presented by Gianluca Chiozzi (ESO)

ICALEPCS 2009 – WEA006

Contents

- ALMA and ACS
- ACS Data Channel
- Data Distribution Service
- The prototype
- Tests
- Conclusion

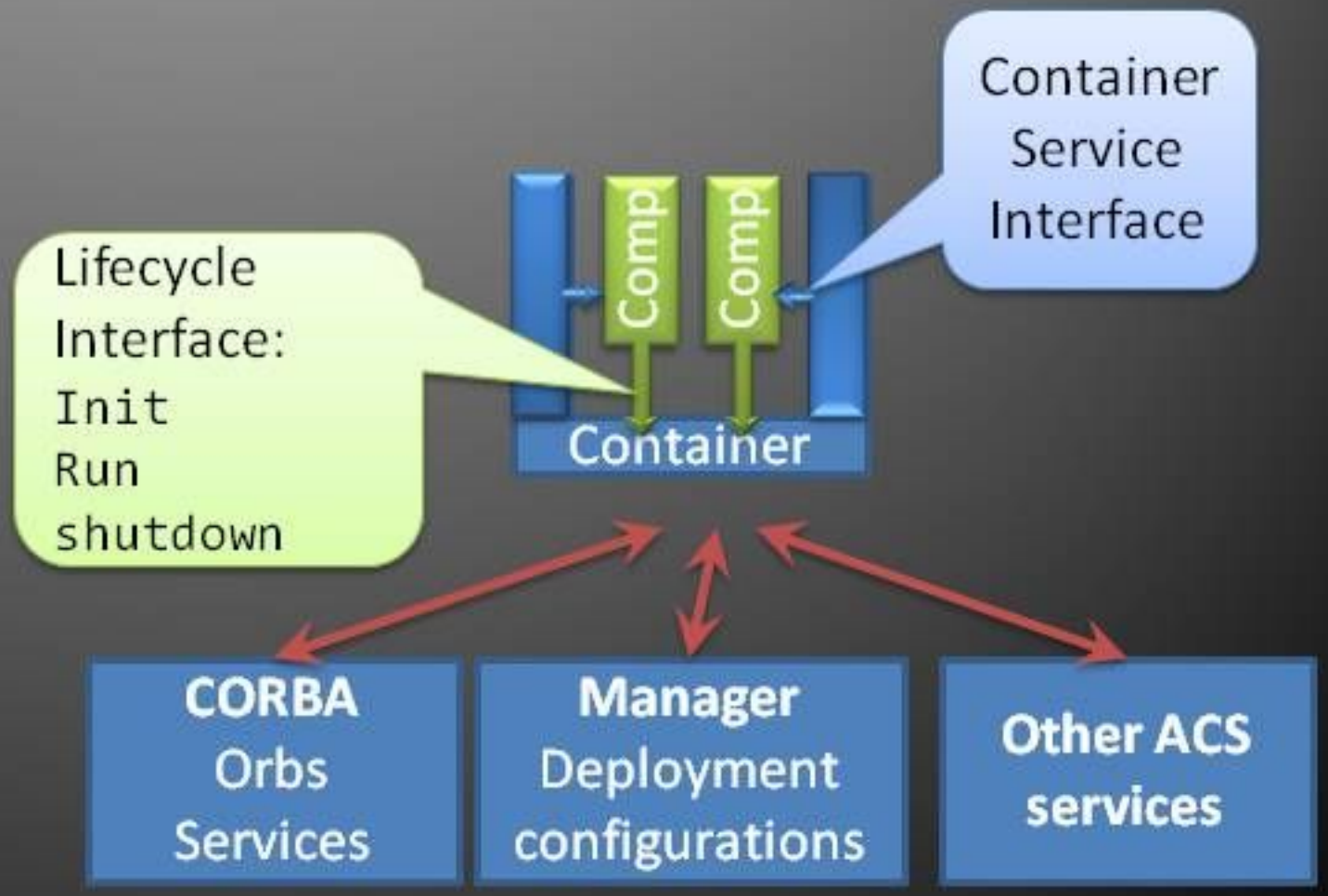
Introduction



ALMA Common Software

ACS is a software infrastructure for the development of distributed systems based on the Component/Container paradigm

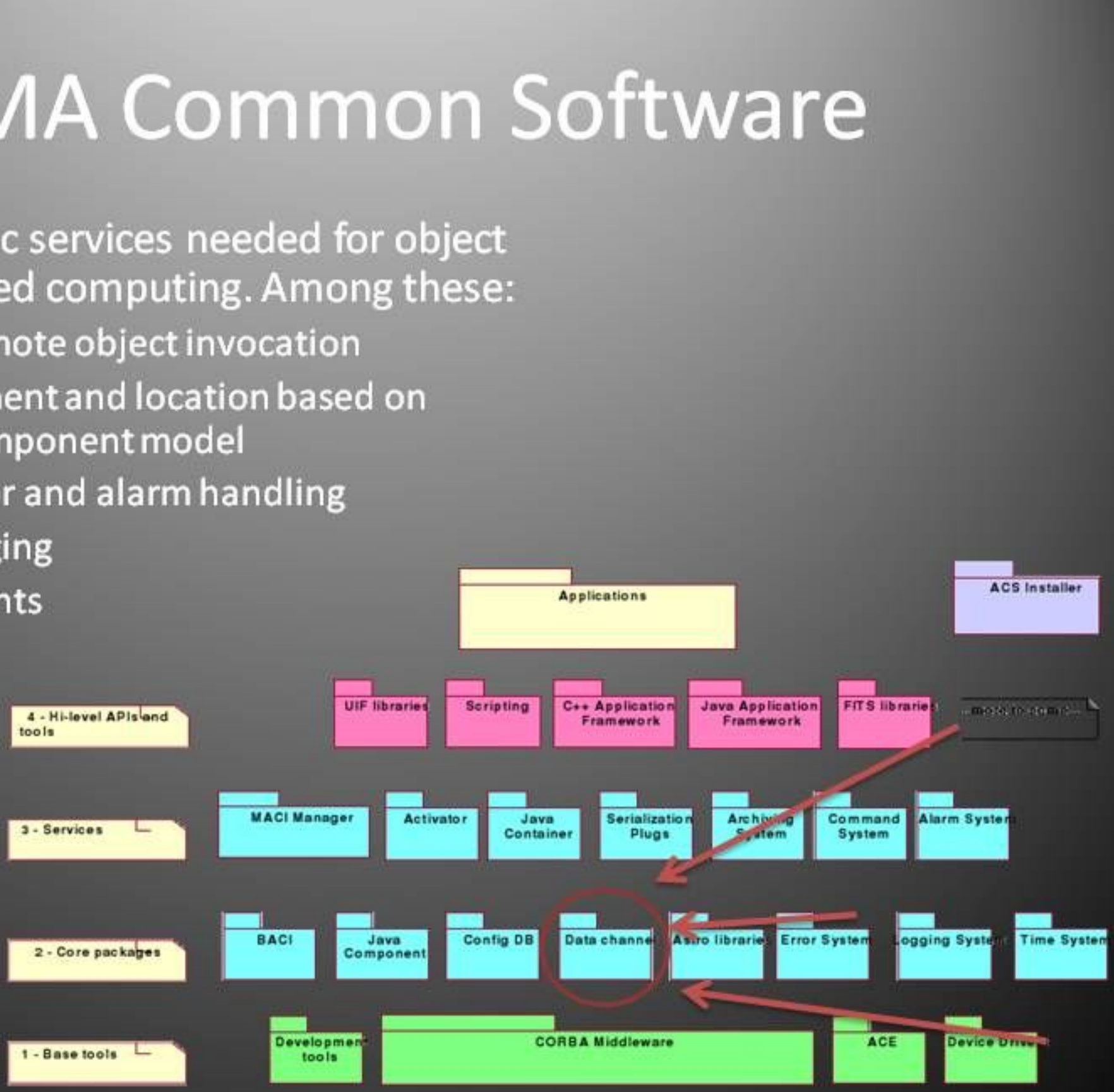
- end-to-end: from data reduction to control applications
- common application framework and design patterns, not just libraries
- well tested software that avoids duplication
- make upgrades and maintenance reasonable
- common development environment and tools
- Open source (LGPL)
- Used by several projects



ALMA Common Software

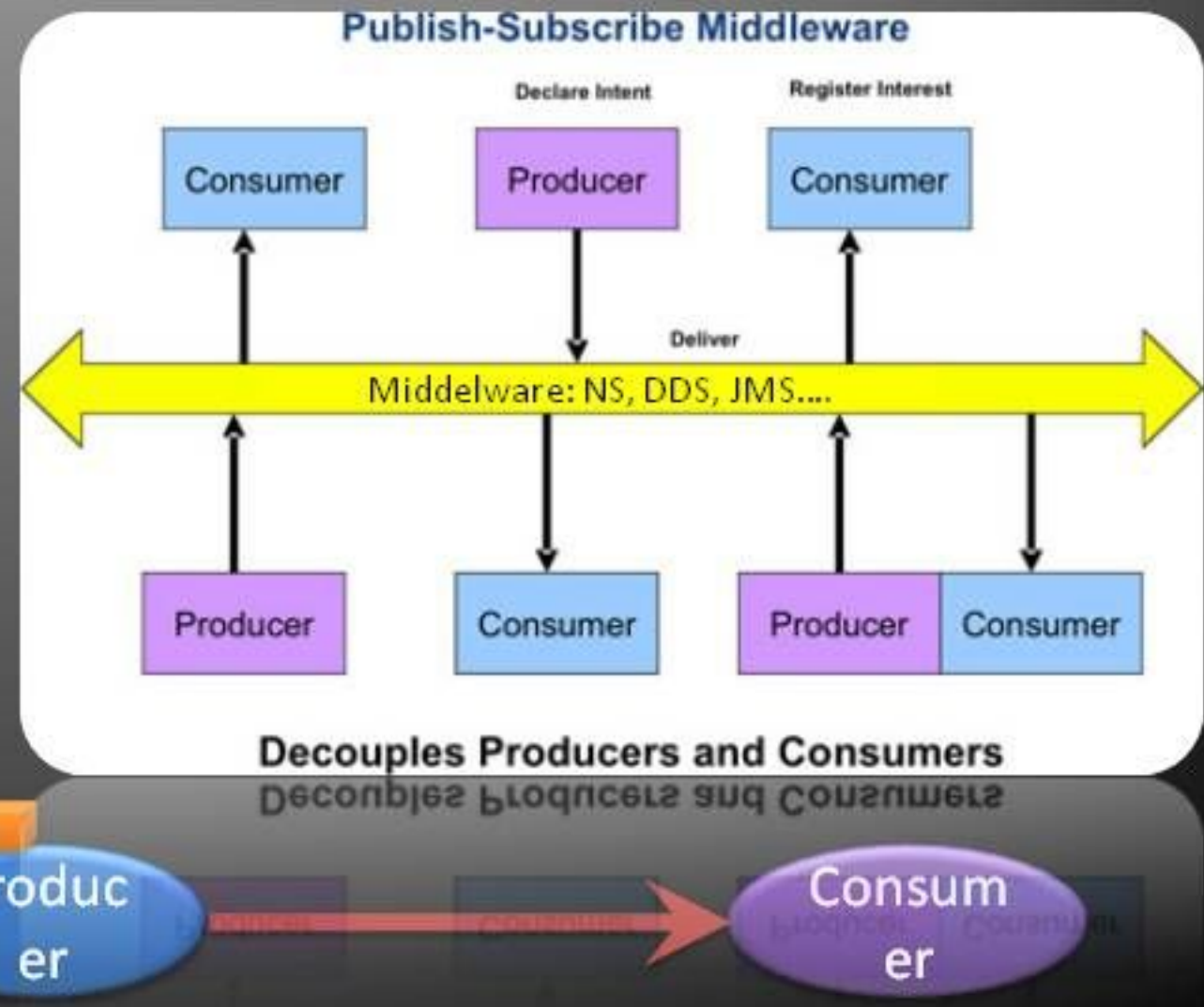
ACS provides the basic services needed for object oriented distributed computing. Among these:

- Transparent remote object invocation
- Object deployment and location based on a container/component model
- Distributed error and alarm handling
- Distributed logging
- Distributed events



ACS Data/Notification Channel

- Publisher/Subscriber mechanism.
- Asynchronous communication.
- Publisher doesn't know about Subscribers.
- Subscriber doesn't know about Publishers.
- Many to many communication.
- Based on CORBA Notify Service.

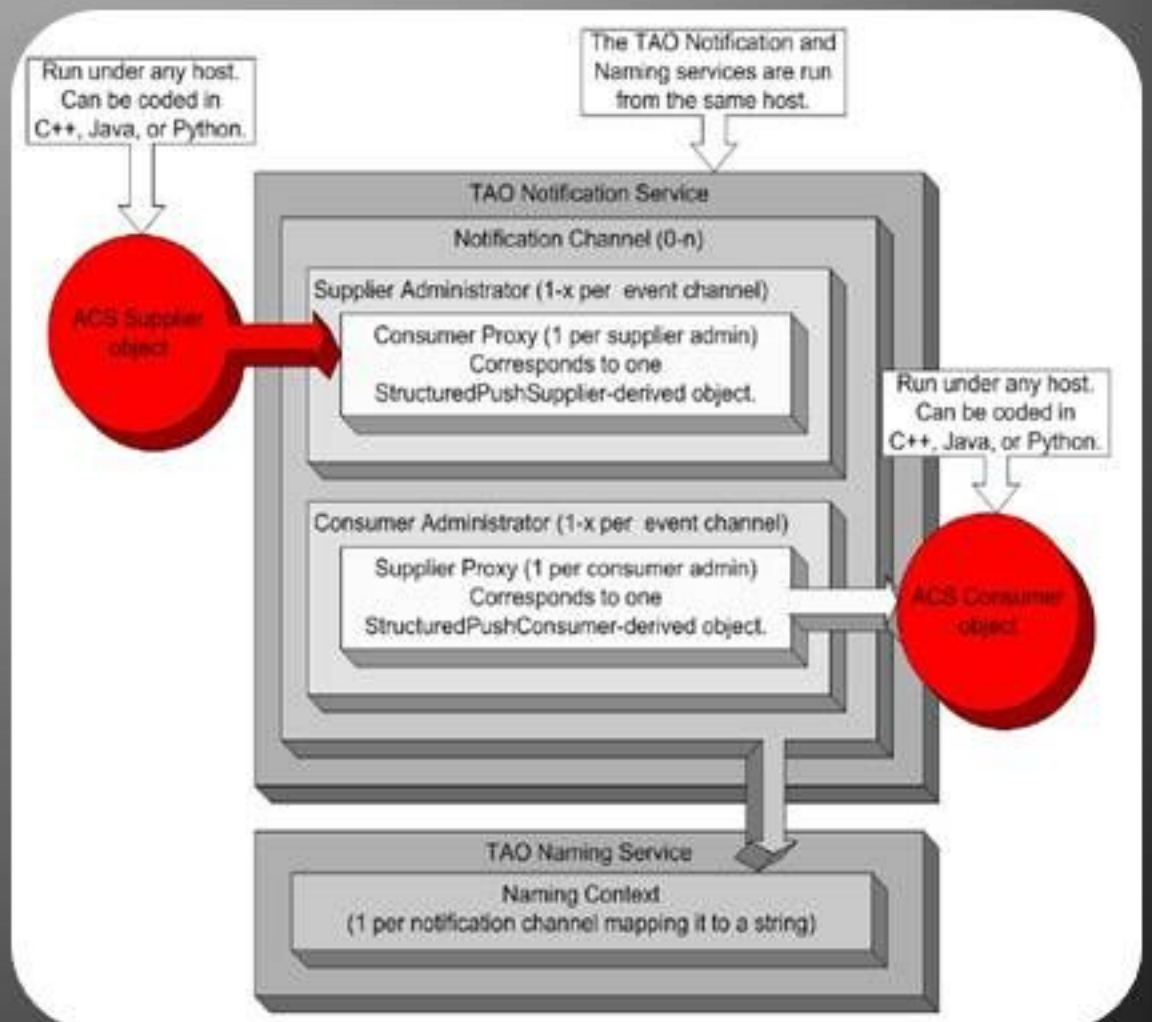


CORBA Notify Service

- Publisher/Subscriber implementation with CORBA.
  - Pull model (Pull Subscriber and Pull Consumer).
  - Push model (Push Subscriber and Push Consumer).
- A broker (Notify Service) is responsible to deliver the messages.
- Message Filtering:
  - Channels.
  - Structured messages.

ACS Data/Notification Channel

- Based on CORBA Notify Service.
- Handles structured messages.
  - Data type is defined in IDL file.
- Support set of Quality of Service properties.
- Uses CORBA Naming Services to make visible the channel.
- Provides:
  - SimpleSupplier class
  - SimpleConsumer class

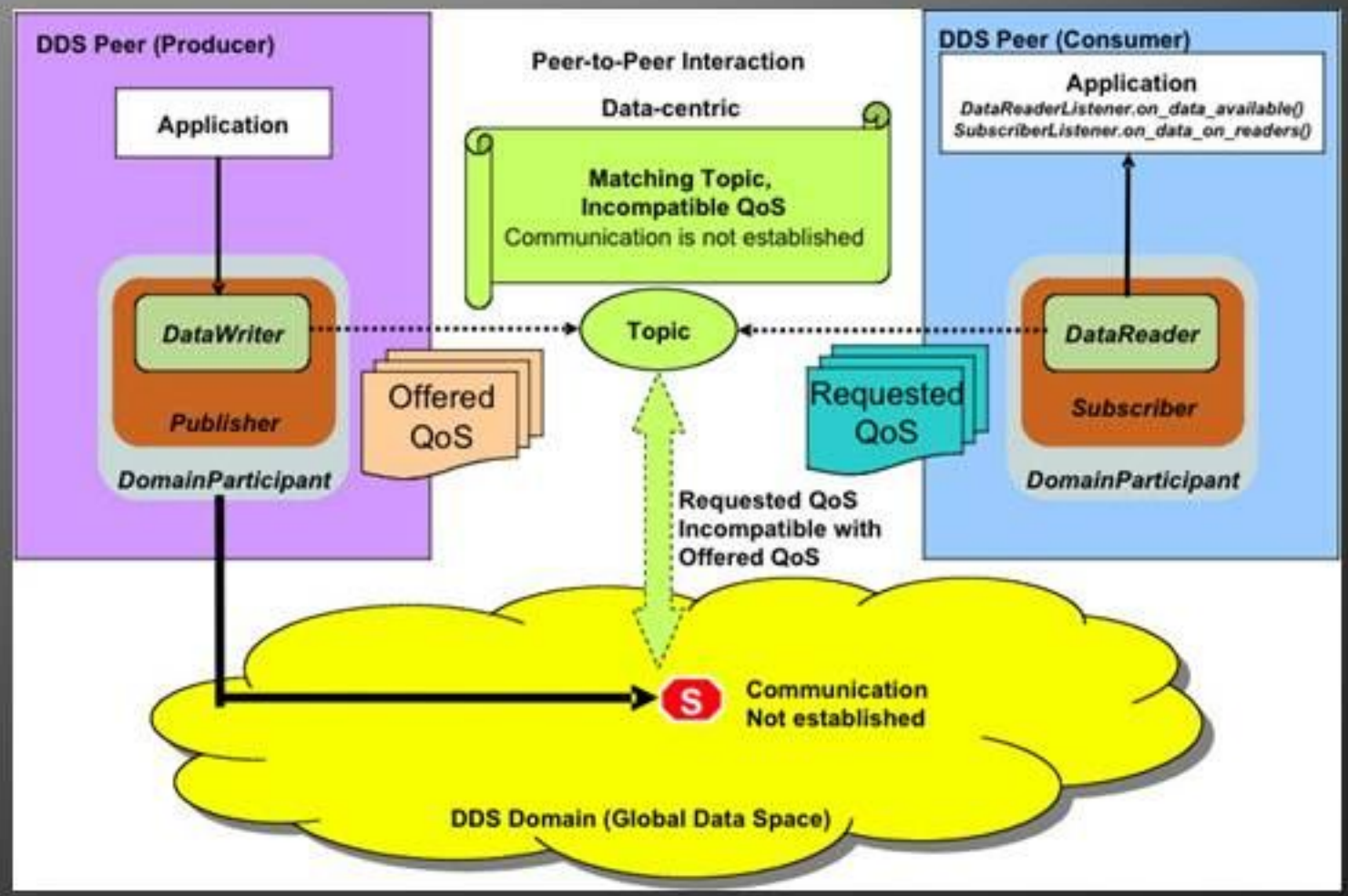


The Problems with NS

- CORBA Notify Service always embeds the message as `Any` CORBA data type:
  - Marshalling uses a lot of CPU time.
  - Length of the message requires a lot of network bandwidth.
- CORBA Notify Service doesn't scale very well.
  - Centralized delivery.
- ACS Notification Channel lacks of *late joining subscriber* feature.

An alternative: Data Distribution Service

- OMG Open Standard (as well as CORBA)
- High performance Publisher/Subscriber specification (focused on Real-Time)
  - Peer to Peer
  - Resource Management through QoS policies.
- Entities:
  - Publisher
    - Data Writer
  - Subscriber
    - Data Reader
  - Topic



Why DDS?

- Real data centric publisher/subscriber.
  Not just events!
- Data Distribution Service can do the same as CORBA Notify Channel.
- But… it could offer more features:
  - Better performance
  - Multicast transport support
  - Can be configured to support *late joining subscriber* feature.
- Would be desirable to provide these features in ACS: *DDS for ACS*.

Objectives of the prototype

- **To compare Data Distribution Service and CORBA Notify Service**, trying to find a common abstraction, considering the functionality offered by the ACS Notification Channel API.
- **Implement an ACS Notification Channel alternative based on OpenDDS (in C++)**, trying to maintain the actual ACS Notification Channel API.
- **Set up a test suite** including some measurement of throughput or other performance indicators.

Data Distribution Service for ACS

- DDS Notification Messaging API
- DDS specification only implements push model.
- Maps:
  - SimpleSupplier in DDSPublisher
  - SimpleConsumer in DDSSubscriber
- Offers a very similar API.
- Hide DDS complexity:
  - Initialization of entities of DDS.
  - Transport selection and initialization of OpenDDS.

VERIFICATION

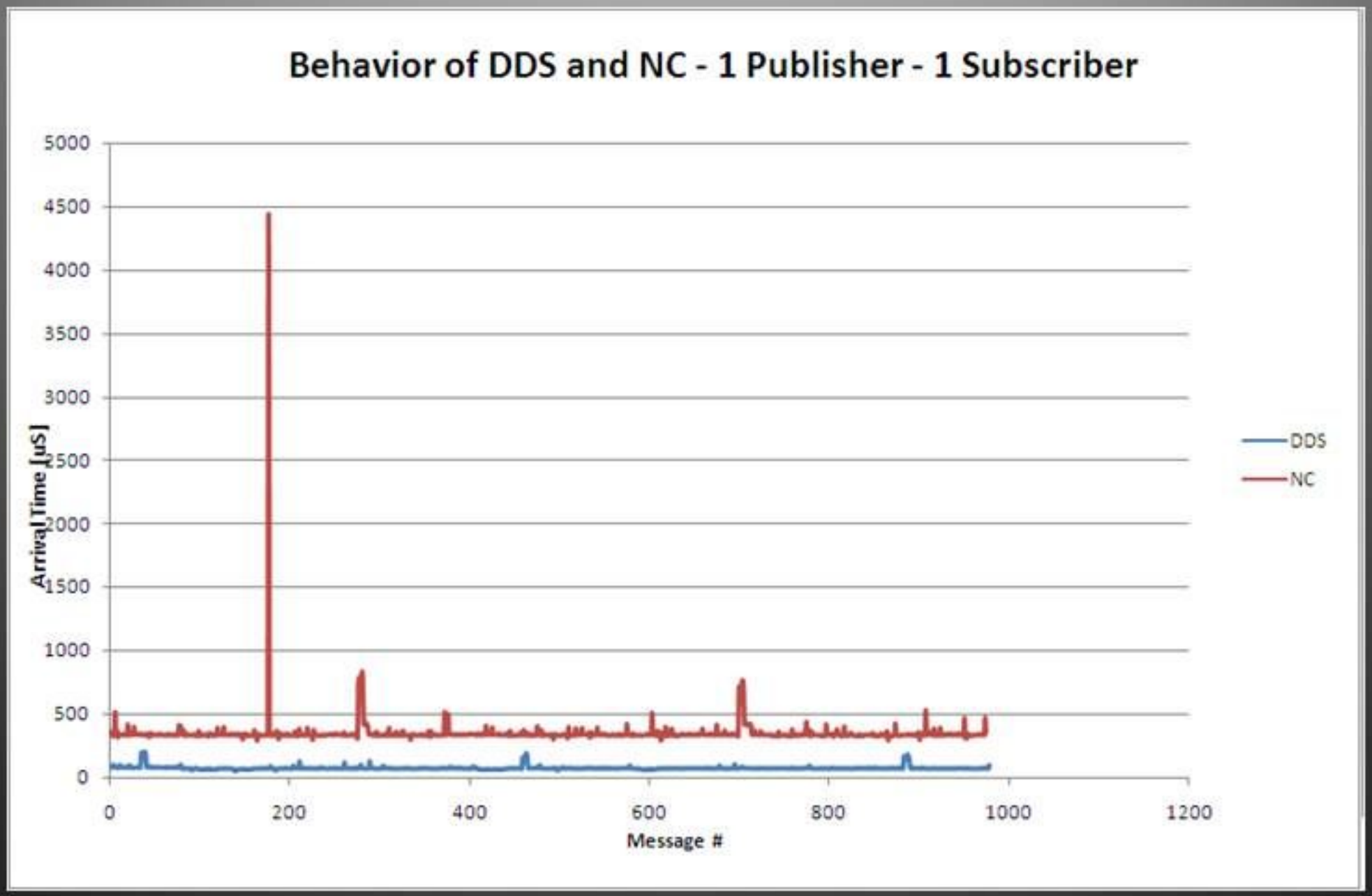
Tests

1. Scalability test
   - 1 Publisher.
   - 1, 10, 20, 30, 50, 60, 75, 100 Subscribers.
   - 1000 messages at 10 Hz.
2. Slow Consumer test
   - 1 Publisher.
   - Several Subscriber and one Slow Subscriber.
3. Throughput test
   - 1 Publisher.
   - 1 Subscriber.
   - Message frequency: as fast as possible.

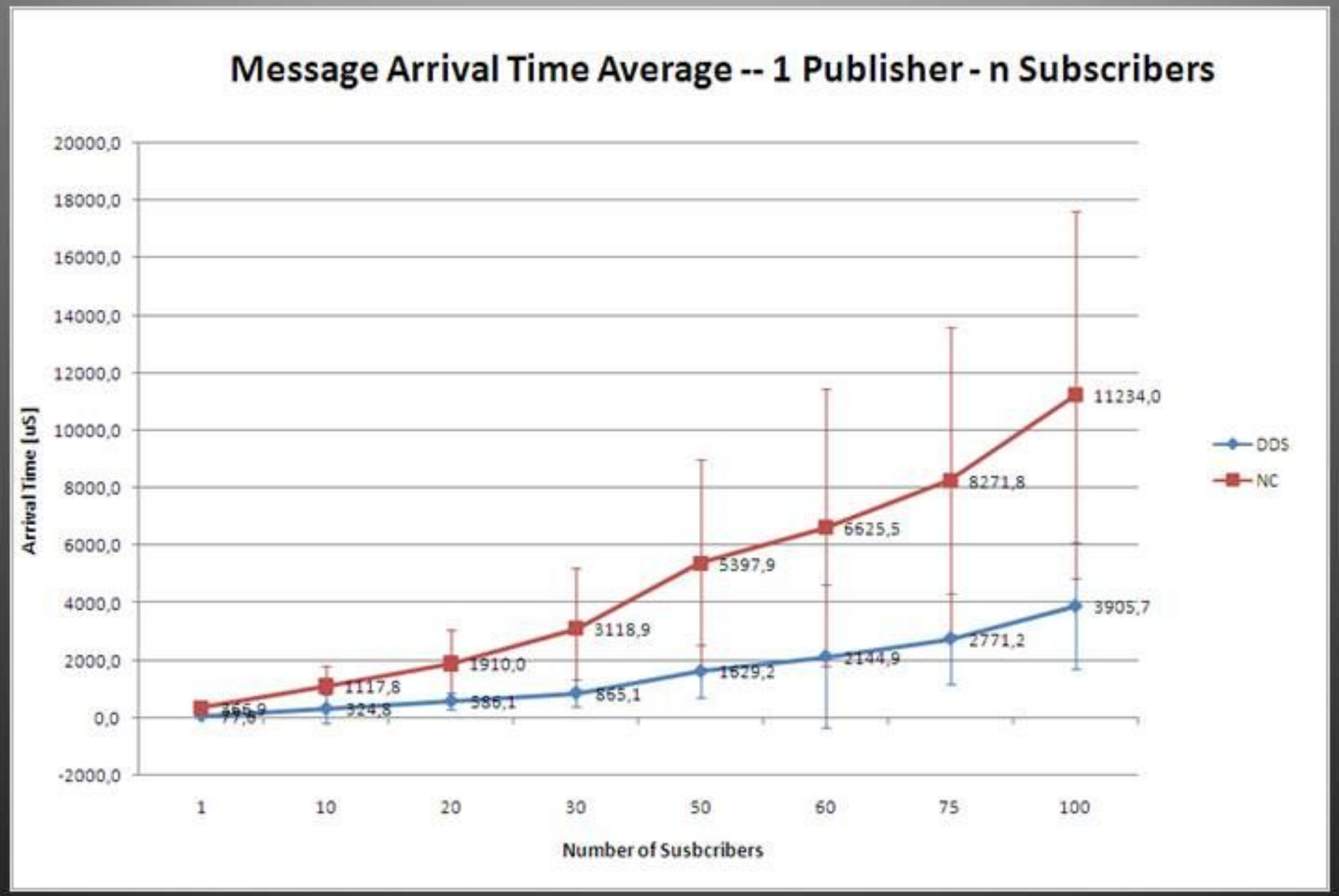
Test Configuration

- ACS 8 Release Candidate running on Scientific Linux 5.2.
- ACE version 5.6.5, provided as part of ACS 8 RC.
- TAO version 1.6.5, provided as part of ACS 8 RC.
- OpenDDS version 1.1

Scalability test results



Scalability test results



Scalability test results: Resource consumption

Program	Memory Usage (MiB)	Number of Threads
DDS Subscriber	**4,8**	**12**
NC Consumer	3,4	2
DDS Publisher (Container)	**55,4**	**216**
NC Supplier (Container)	3,9	7
DCPSInfoRepo	**73**	**206**
NotifyService	68,9	102

Throughput test

- Container crashes for memory exhaustion queuing undelivered messages with both implementations.
- Some values:

	DDS	NS
Message Received	~ 80.000	~ 230.000
Maximum Latency [uS]	~ 2.500.000	~ 35.000.000

- The crash can be avoided in DDS
  - Set correctly the QoS properties (History and Resources).

CONCLUSIONS

Conclusion

- ACS has now a prototype DDS replacement for the CORBA Notify Service.
- We have performed a comparison between the two implementations:
  + Good performance and scalability
  + Good QoS options and granularity
  - A lot of memory
  - A lot of threads
  - Interoperability and portability still limited
- The prototype is used for the E-ELT evaluation of ACS.
- Future Work
  - Get from prototype to “production quality”
  - Port all other ACS Services now using NS to DDS.
  - Test and compare with RTI DDS and OpenSplice
  - Test Multicast protocols.

QUESTIONS?

For more details:

http://www.eso.org/projects/alma/develop/acs

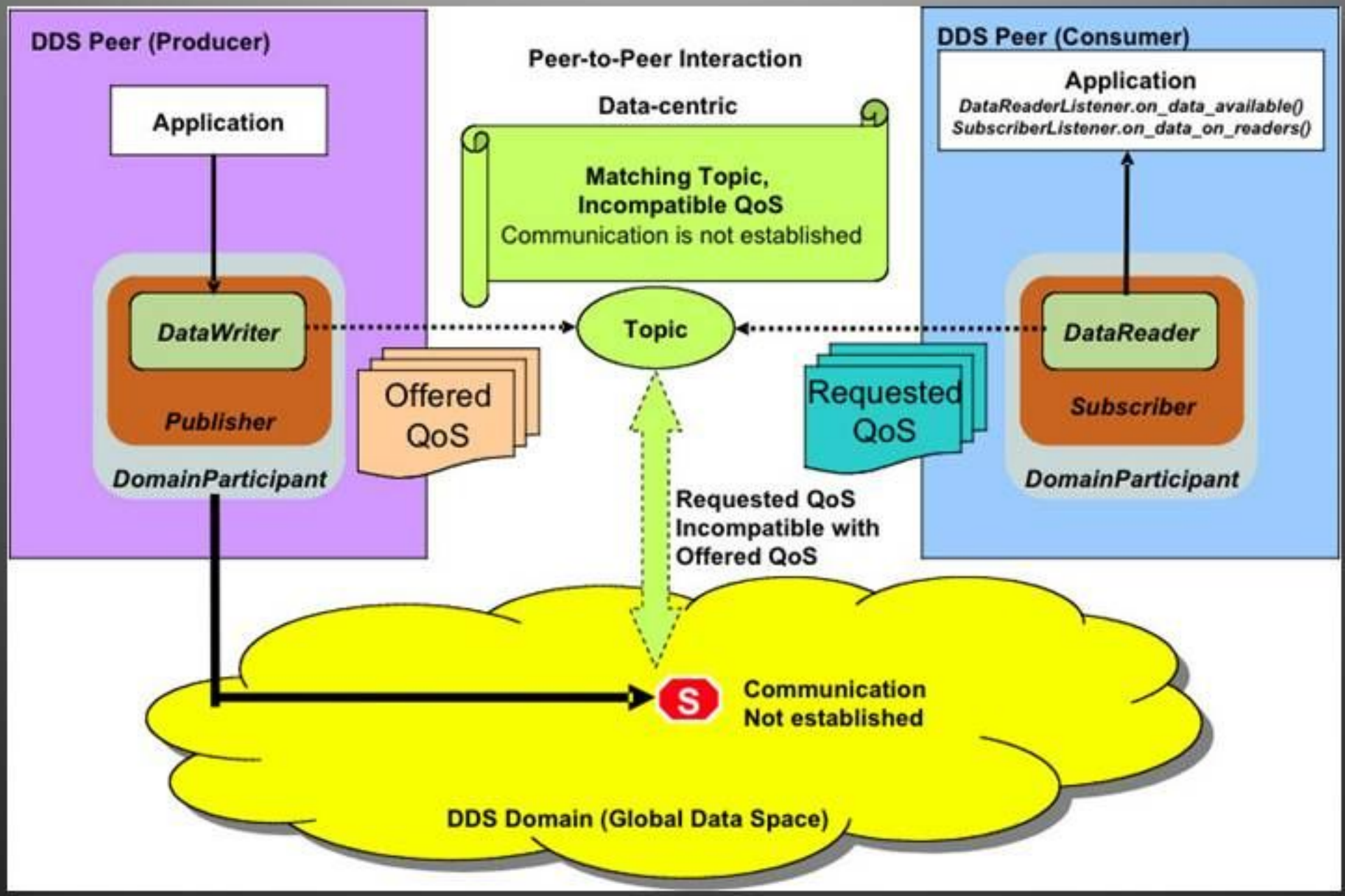
Next ACS workshop:

UTFSM Valparaiso, Chile
November 2009

This work was supported by the ALMA-Conicyt Fund #31060008

EXTRA SLIDES

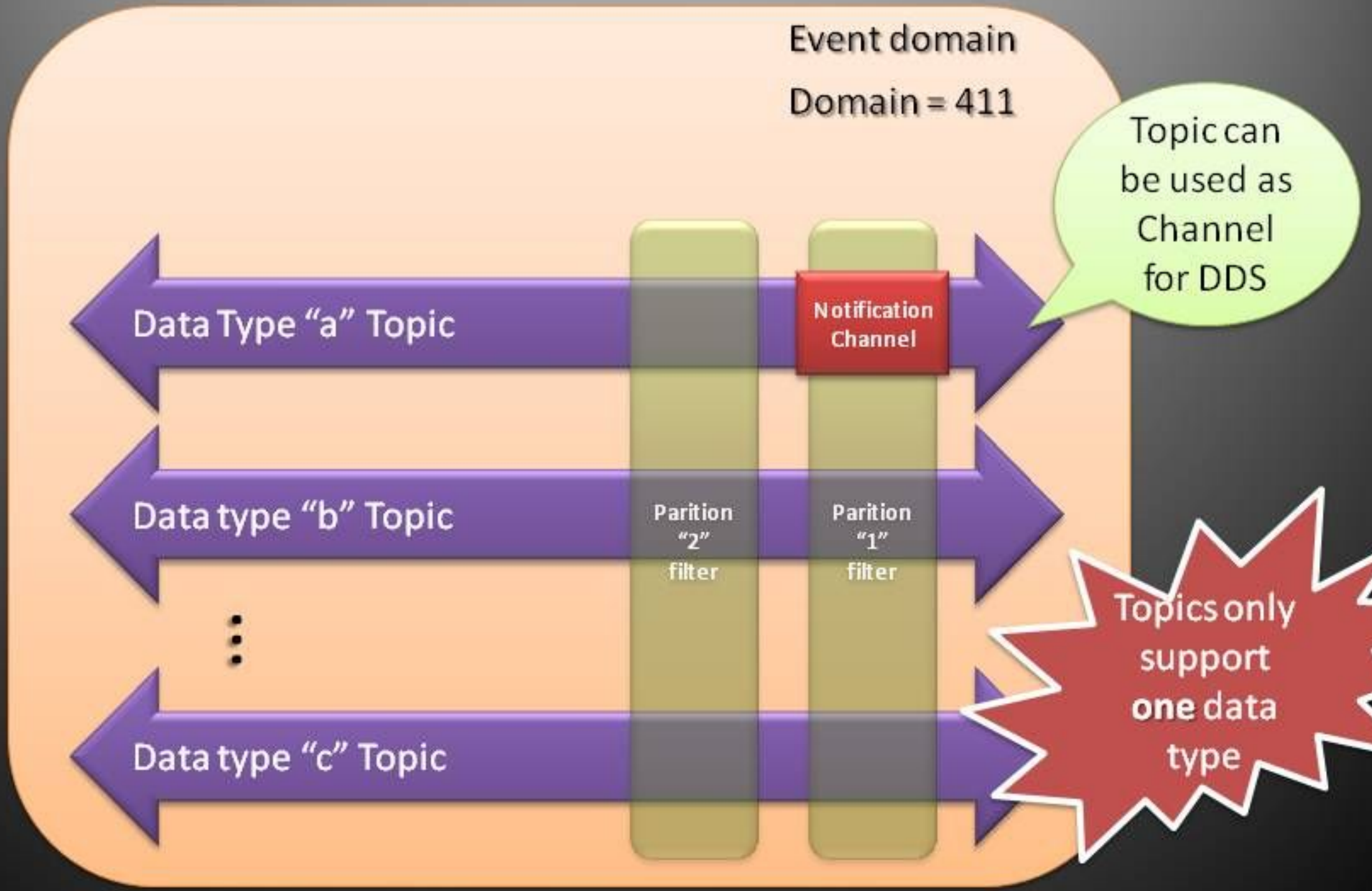
Data Distribution Service



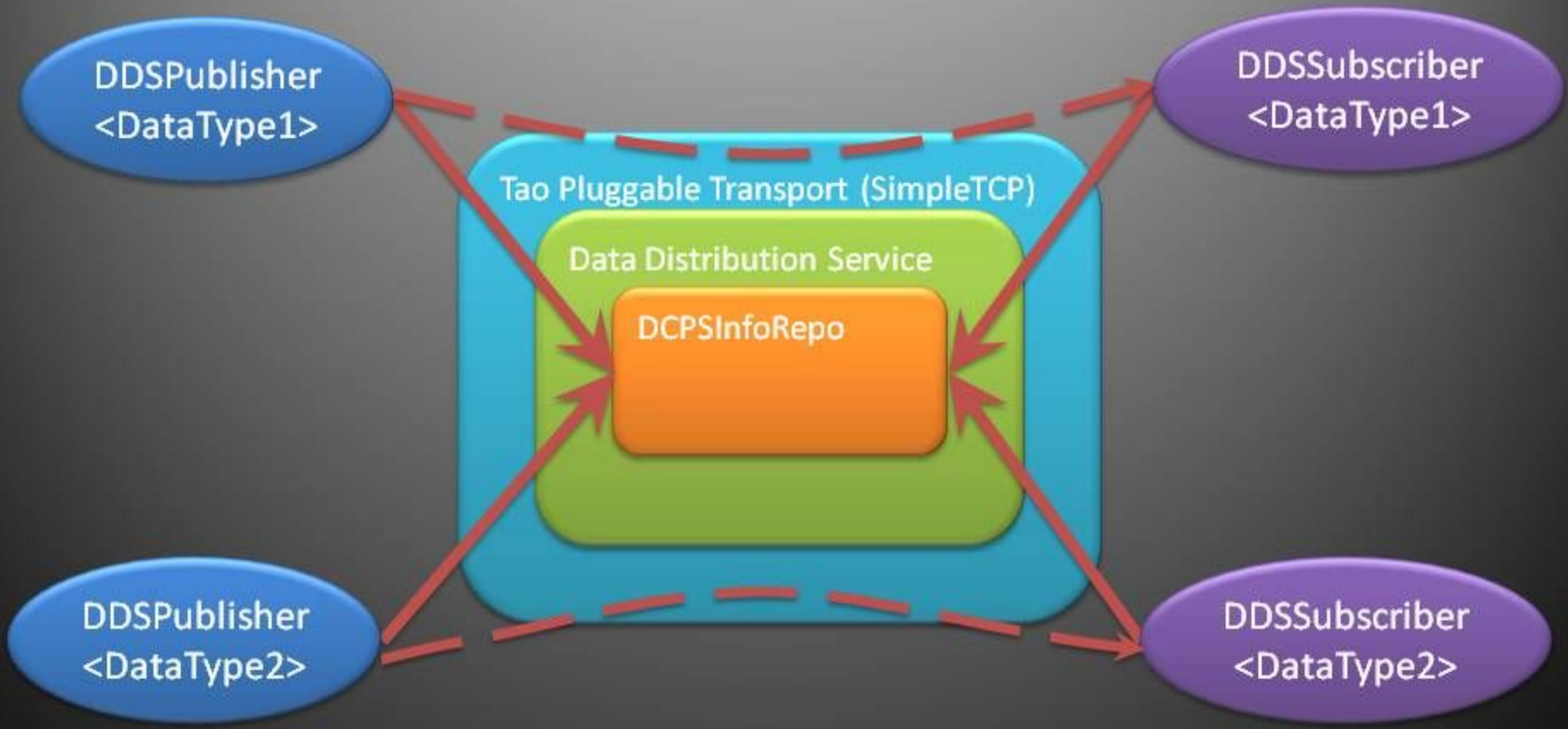
ACS Notification Channel features

- Hide as much as possible the CORBA complexity.
- Have three levels of message filtering
  - Event Domain
  - Event Type
  - Channel Name
- Offers two simple classes:
  - `SimpleSupplier`, Publisher class
    - `void publishData(T data)`
  - `SimpleConsumer`, Subscriber class
    - `void consumerReady()`

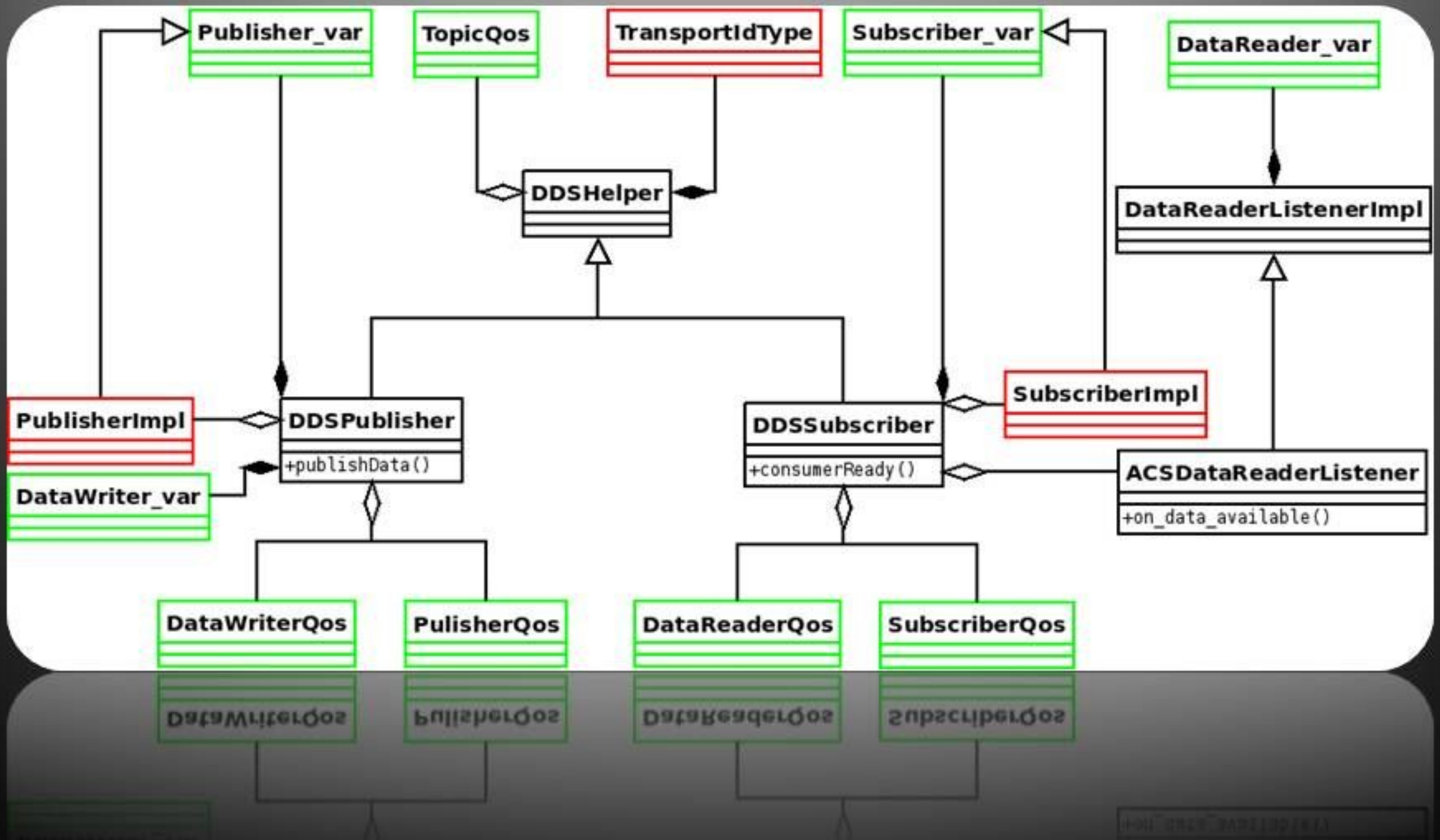
DDS Conceptual Mapping



DDS Example Deployment



Solution Proposal: Class model



Data Definition: IDL file

```
module DDS_SIMPLE_EXAMPLE{
   const string CHANNEL_NAME = "simple_example";

#pragma DCPS_DATA_TYPE
"DDS_SIMPLE_EXAMPLE::simpleMessage"
struct simpleMessage{
    unsigned long seqnum;
};
```

Publisher

DDS

```
void SimpleExampleDDSImpl::sendMessage()
{
    pub_p = new acsPublisher(
            DDS_SIMPLE_EXAMPLE::CHANNEL_NAME, component, [NC/DDS]);

    DDS_SIMPLE_EXAMPLE::simpleMessage m;
    m.seqnum=1;

    PUBLISH_DATA(pub_p, DDS_SIMPLE_EXAMPLE::simpleMessage, m);

    sleep(1);
    pub_p->disconnect();
    delete pub_p;
}
```

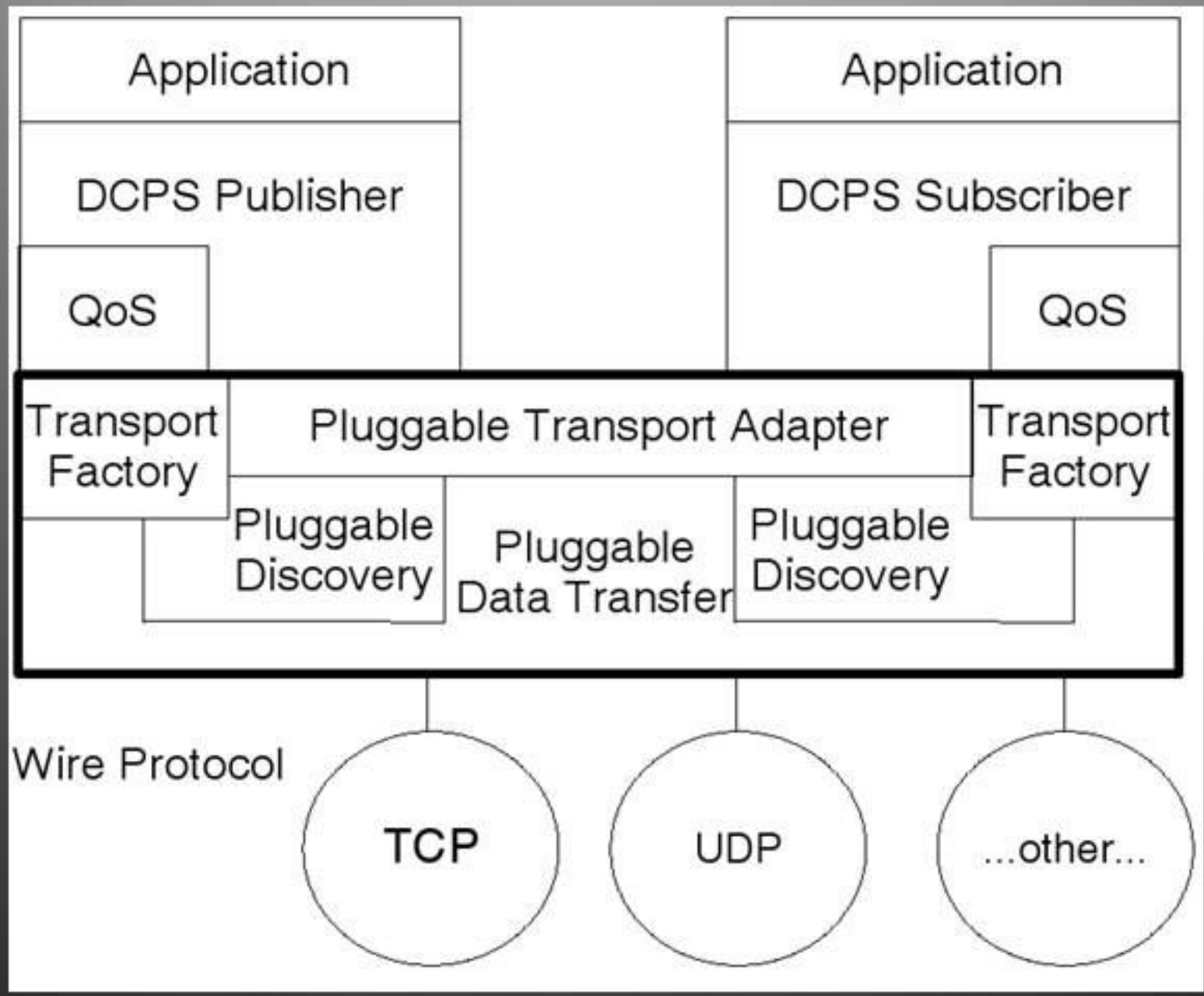
Subscriber

```
void handlerFunction(DDS_SIMPLE_EXAMPLE::simpleMessage m, void *other)
{
    std::cout << "Arrived message" << std::endl;
}

Void main()
{
….
    ddsnc::DDSSubscriber *sub_p=0;

    ACS_NEW_DDS_SUBSCRIBER(sub_p,
                           DDS_SIMPLE_EXAMPLE::simpleMessage,
                           DDS_SIMPLE_EXAMPLE::CHANNEL_NAME,
                           &handlerFunction, (void *)0);

    sub_p->consumerReady();

    ACE_Time_Value tv(100);
    client.run(tv);

    sub_p->disconnect();
    delete sub_p;
}
```

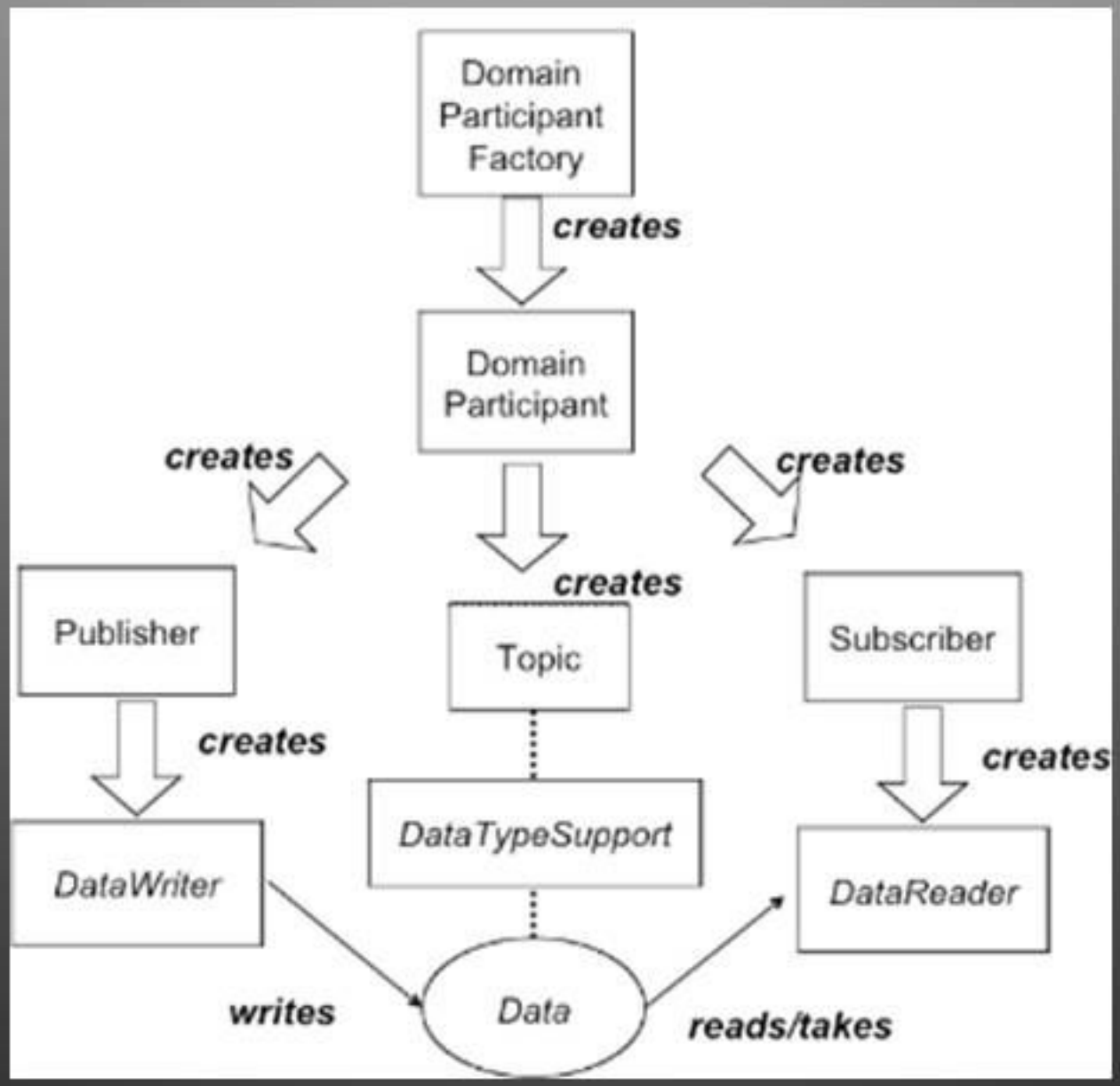
Slow Consumer test results

- In ACS NC Notify Service memory usage raises
  - Reliable QoS policy retains the messages not delivered.
- DDS have the same behavior
  - But it can be solved setting correctly the QoS properties (History and Resources)
- In DDS implementation the “normal” subscribers don’t have delays.

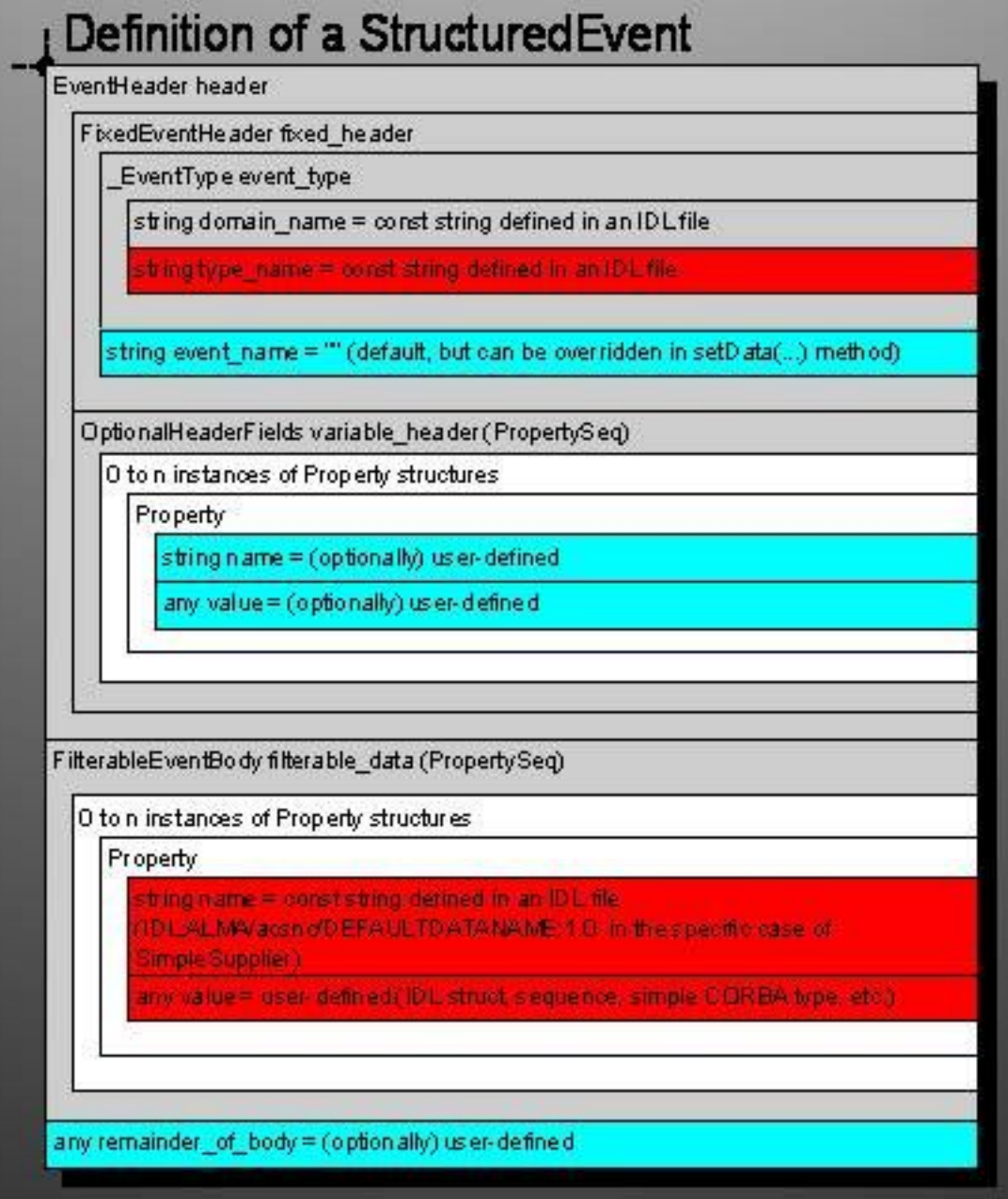
OpenDDS Architecture



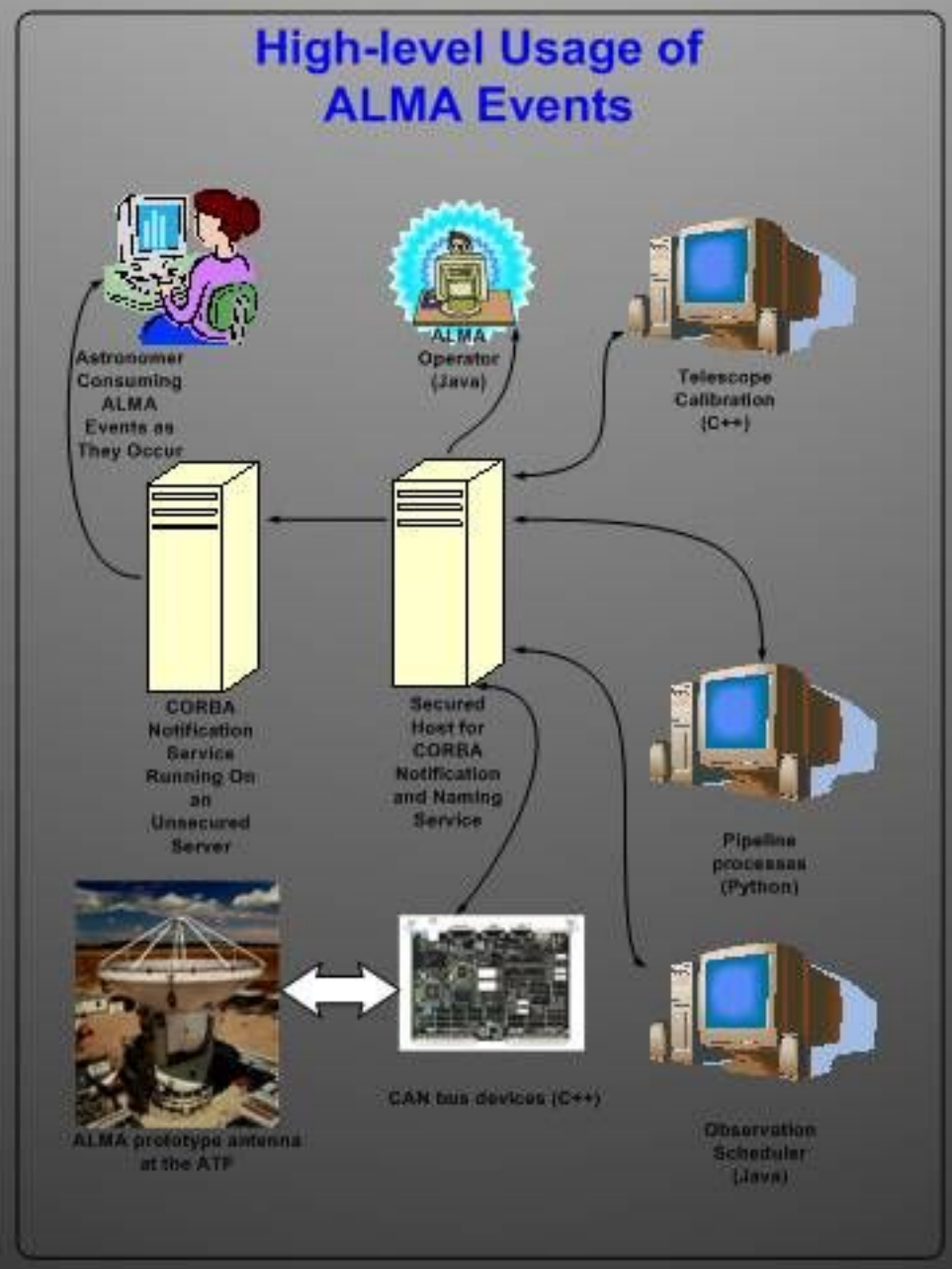
OpenDDS programming



ACS NC Structured Message



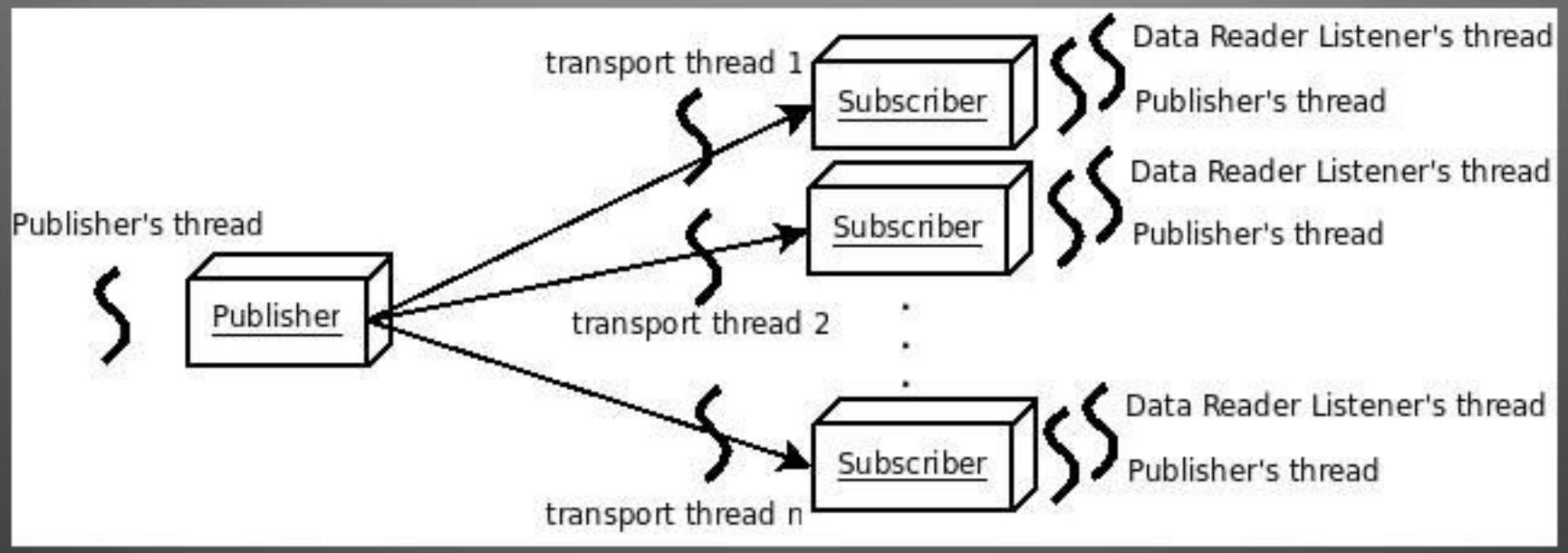
Usage of ACS Notification Channel



Technical Details

- OpenDDS availability as part of ACS.
- Compilation of OpenDDS with ACS.
- Modifications done to ACS Makefile
  - OpenDDS processing.
    - dcps_ts.pl
    - Generated extra classes for type support.

Threads Management



Problems, Solutions and Alternatives

- DDS CORBA Compatibility
- Participants parameters required by OpenDDS
  - DCPSInfoRepo Location
  - Transport Configuration
  - Configuration File
- Factories
  - Participant
  - Transport

Known Problems

- Threads and Container
  - The participant factory and the transport factory are threads that cannot be stopped.
- Resource consumption (threads)
  - For each participant in the connection, OpenDDS creates a new thread to establish the communication between the others participants.
  - Requires a lot of memory in comparison of NC.
- Occasionally container crash
  - When a subscriber finish after the component is deactivated

Acknowledgements

- *ESO and the ALMA-Conicyt Fund #31060008*
- *Heiko Sommer for his dedication and help guiding this work.*
- *Matías Mora and Rodrigo Tobar for their support.*
- *Thesis advisors: Horst von Brand and Javier Cañas.*
- *My family, my friends and the Computer Systems Research Group.*