Presentation

Agenda

OMG Data-Distribution Service (DDS): Architectural Overview

Gerardo Pardo-Castellote
Real-Time Innovations, Inc. (RTI)
Phone: 1-408-734-4200, x106
Email: pardo@rti.com

Topic Areas Software Architectures, Reusability, Scalability, and Standards
Middleware Libraries and Application Programming Interfaces
Fault-Tolerant Hardware and Software Techniques

Summary

The OMG Data-Distribution Service (DDS) is a new specification for publish-subscribe data-distribution systems. The purpose of the specification is to provide a common application-level interface that clearly defines the data-distribution service. The specification describes the service using UML, providing a platform-independent model that can then be mapped into a variety of concrete platforms and programming languages.

This paper introduces the OMG DDS specification, describes the main aspects of the model, compares it with related technologies, and gives examples of the communication scenarios it supports.

This paper and presentation will clearly explain the important differences between **data-centric publish-subscribe** and **object-centric client-server** (e.g. CORBA) communications, along with the applicability of each for real-time systems.

The OMG DDS attempts to unify the common practice of several existing implementations enumerating and providing formal definitions for the Quality of Service (QoS) settings that can be used to configure the service.

Publish-subscribe networking is a key component of the Navy Open Systems Architecture (**Navy OA**) initiative. This talk will also highlight practical publish-subscribe implementations in Navy systems such as LPD 17, SSDS, and DD(X).

Background

The goal of the DDS specification is to facilitate the efficient distribution of data in a distributed system. Participants using DDS can "read" and "write" data efficiently and naturally with a typed interface. Underneath, the DDS middleware will distribute the data so that each reading participant can access the "most-current" values. In effect, the service creates a global "data space" that any participant can read and write. It also creates a name space to allow participants to find and share objects.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **01 FEB 2005**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **OMG Data-Distribution Service (DDS)**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Real-Time Innovations, Inc. (RTI)**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001742, HPEC-7 Volume 1, Proceedings of the Eighth Annual High Performance Embedded Computing (HPEC) Workshops, 28-30 September 2004. , The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**28**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

DDS targets real-time systems; the API and QoS are chosen to balance predictable behavior and implementation efficiency/performance. We will note some of these tradeoffs in this paper.

Data-Centric versus Object-Centric communications models

Central to understanding the need for this new standard is an examination of the fundamental architectural differences between a “data-centric” and “object-centric” view of information communicated in a distributed real-time system.

DDS provides a natural counterpoint to the existing well-known CORBA model in which method invocations on remote objects are accessed through an interface defined in the Interface Descriptor Language (IDL). With CORBA, data is communicated indirectly through arguments in the method invocations or through their return values.

However, in many real-time applications the communications pattern is often modeled as pure data-centric exchange where applications publish supply or stream) “data” which is then available to the remote applications that are interested in it. Of primary concern is the efficient distribution of data with minimal overhead and the need to scale to hundreds or thousands of subscribers in a robust, fault-tolerant manner. These types of applications can be found in C4I systems, distributed control and simulation, telecom equipment control, and network management.

Comparison to Distributed Shared Memory

Additional requirements of many real-time applications include the need to control QoS properties that affect the predictability, overhead, and resources used. Distributed shared memory is a classic model that provides data-centric exchanges. However, this model is particularly difficult and “unnatural” to implement efficiently over the Internet.

Therefore, another model, the Data-Centric Publish-Subscribe (DCPS) model, has become popular in many real-time applications. While there are several commercial and in-house developments providing this type of facility, to date, there have been no general-purpose data-distribution standards. As a result, no common models directly support a data-centric system for information exchange.

The OMG Data-Distribution Service (DDS) is an attempt to solve this situation. The specification also defines the operations and QoS attributes each of these objects supports and the interfaces an application can use to be notified of changes to the data or wait for specific changes to occur.

Comparison to existing OMG Notification Service

This paper will examine the fact that, while it is theoretically possible for an application developer to use the OMG Notification Service to propagate the changes to data structures to provide the functionality of the DDS, doing this would be significantly complex because the

Notification Service does not have a concept of data objects or data-object instances nor does it have a concept of state coherence.

Comparison to existing High-Level Architecture (HLA) Run-Time Infrastructure (RTI)

HLA, also known as the OMG Distributed Simulation Facility, is a standard from both IEEE and OMG. It describes a data-centric publish-subscribe facility and a data model. The OMG specification is an IDL-only specification and can be mapped on top of multiple transports. The specification address some of the requirements of data-centric publish subscribe: the application uses a publish-subscribe interface to interact with the middleware, and it includes a data model and supports content-based subscriptions.

However, the HLA data model supports a specialization hierarchy, but not an aggregation hierarchy. The set of types defined cannot evolve over time. Moreover, the data elements themselves are un-typed and un-marshaled (they are plain sequences of octets). HLA also offers no generic QoS facilities.

Applications

This paper will describe the successful implementation of data-centric publish-subscribe communications in distributed modeling and simulation (M&S) as well as deployed Navy systems (pending release permissions). The presentation can include examples (depending on audience interest and familiarity) such as:

Ship:	Raytheon/Lockheed Martin LPD-17 Program
Ground:	CLIP/LINK tactical communications Program
Air:	F-35 JSF EW Subsystem
Space:	NASA Robonaut Program

DDS

Data Distribution Service

Gerardo Pardo-Castellote, Ph.D.
Real-Time Innovations, Inc.

DDS Standard

Data Distribution Service for Real-Time Systems

- Adopted in June 2003
- Finalized in June 2004
- Joint submission (RTI, THALES, MITRE, OIS)
- API specification for Data-Centric Publish-Subscribe communication for distributed real-time systems.

RTI's role

- Member of OMG since 2000
- Co-authors of the original DDS RFP
- Co-authors of the DDS specification adopted in June 2003
- Chair of the DDS Finalization Task Force completed March 2004
- Chair of the DDS Revision Task Force
- Providers of a COTS implementation of the specification (NDDS.4.0)

OMG Middleware standards

CORBA

Distributed object

- Client/server
- Remote method calls
- Reliable transport

Best for

- Remote command processing
- File transfer
- Synchronous transactions

DDS

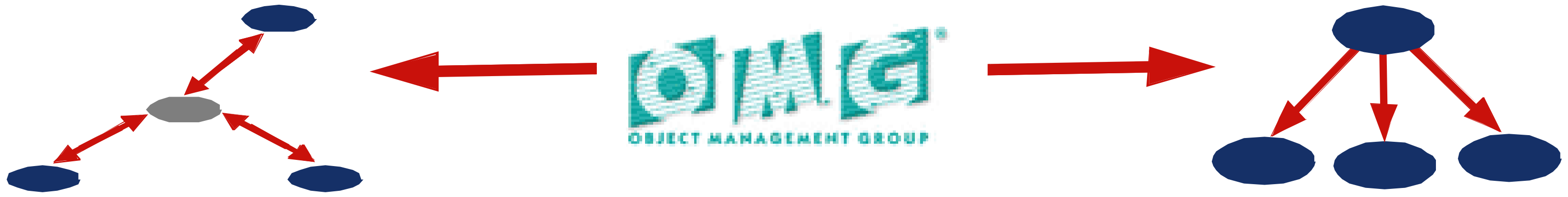
Distributed data

- Publish/subscribe
- Multicast data
- Configurable QoS

Best for

- Quick dissemination to many nodes
- Dynamic nets
- Flexible delivery requirements

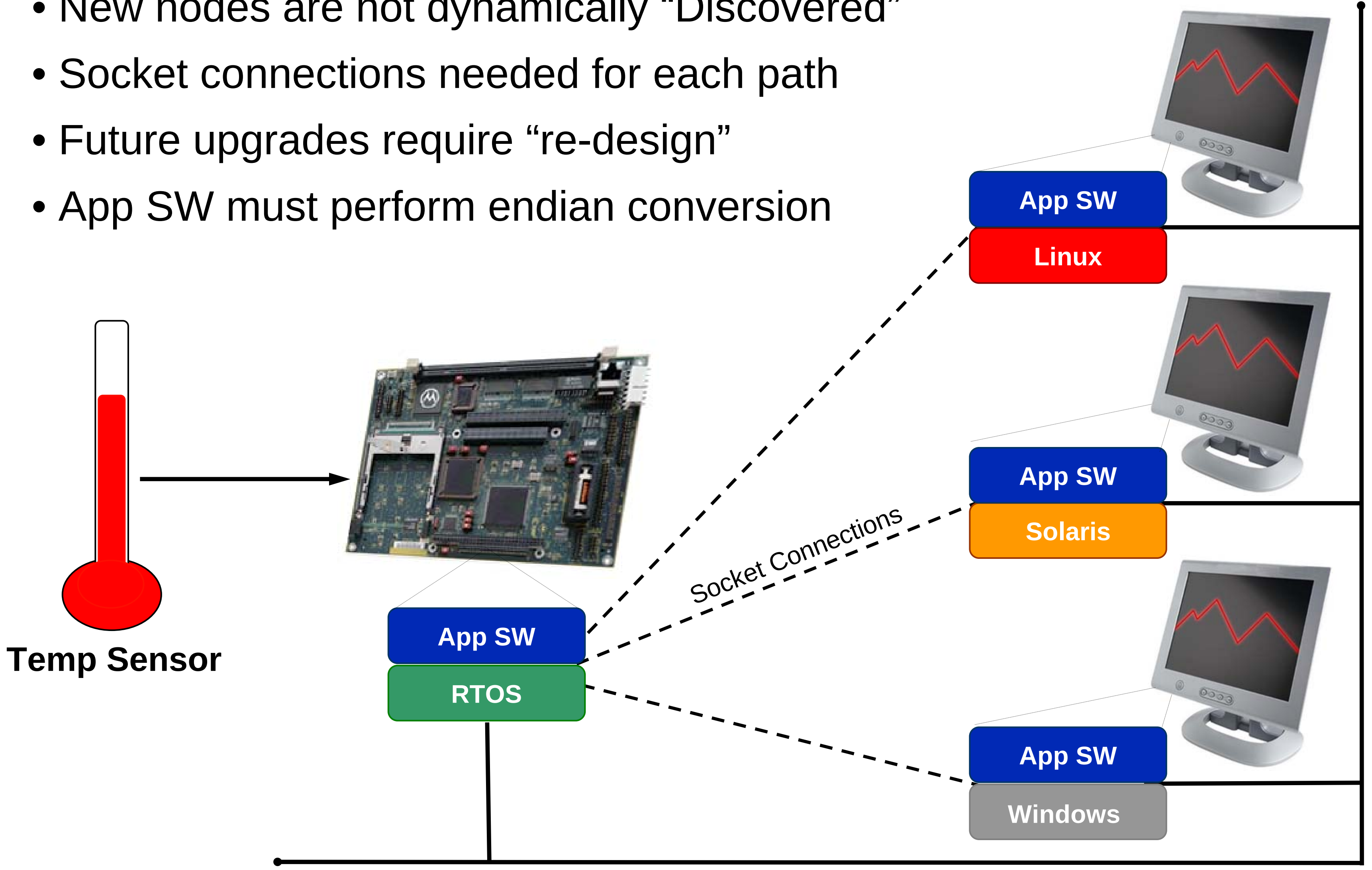
DDS and CORBA address different needs



Complex systems often need both…

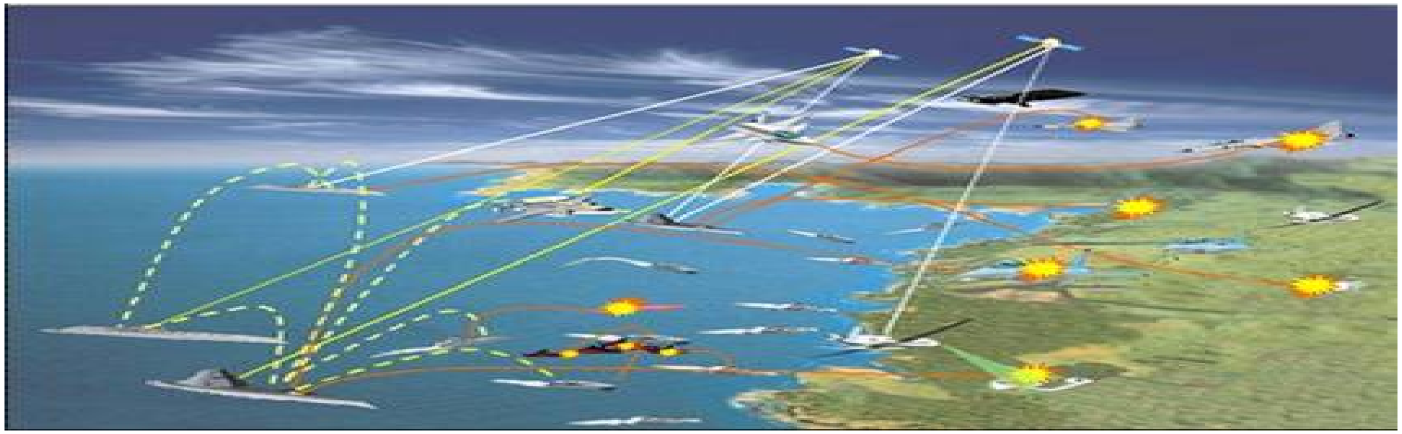
More Complex Distributed Application

- New nodes are not dynamically “Discovered”
- Socket connections needed for each path
- Future upgrades require “re-design”
- App SW must perform endian conversion



The net-centric vision

- Vision for “net-centric applications”
- Total access to information for real-time applications
- This vision is enabled by the internet and related network technologies
- Challenge:
  - **“Provide the right information at the right place at the right time… no matter what.”**



Challenges: Factors driving DDS

Need for speed

- Large networks, multicast
- High data rates
- Natural asynchrony
- Tight latency requirements
- Continuously-refreshed data

Complex data flows

- Controlled QoS: rates, reliability, bandwidth
- Per-node, or per-stream differences
- Varied transports (incl. Unreliable e.g. wireless)

Dynamic configurations

- Fast location transparency

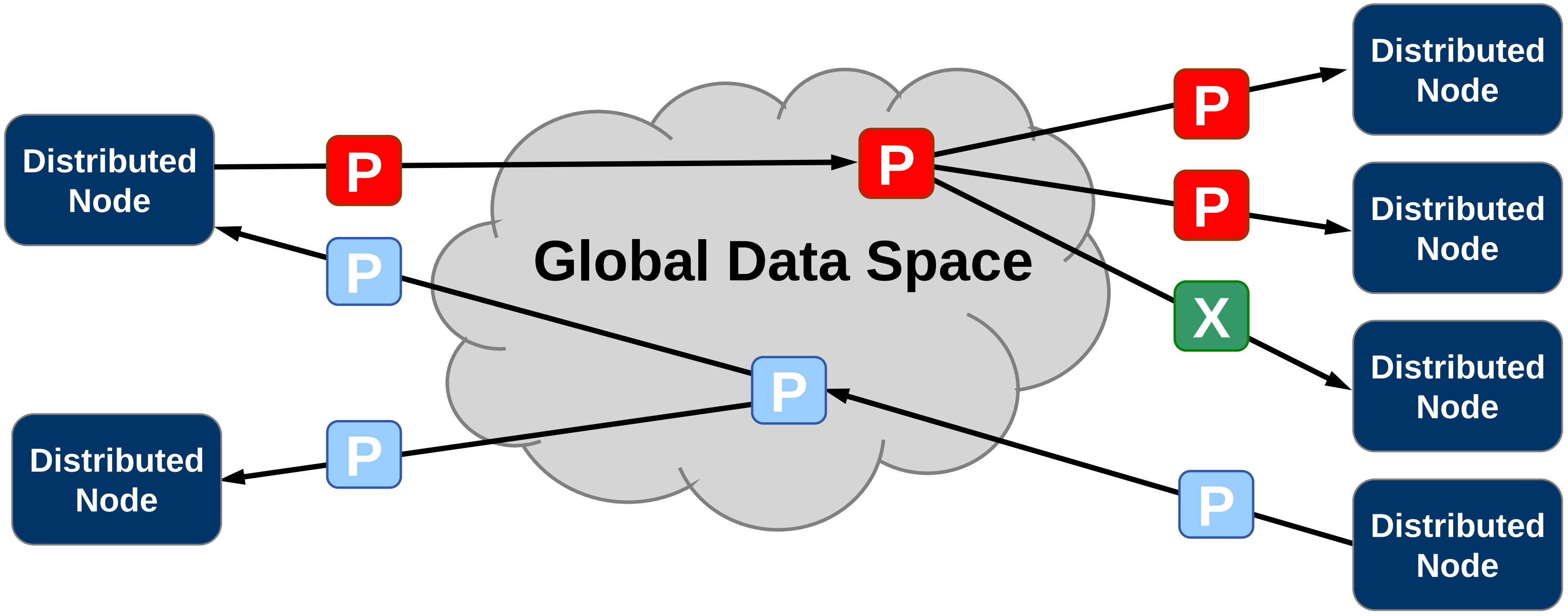
Fault tolerance

- No single-points of failure
- Transparent failover

DDS

Provides a "Global Data Space" that is accessible to all interested applications.

- Data objects addressed by ***Topic*** and ***Key***
- Subscriptions are ***decoupled*** from Publications
- Contracts established by means of ***QoS***
- Automatic ***discovery*** and configuration

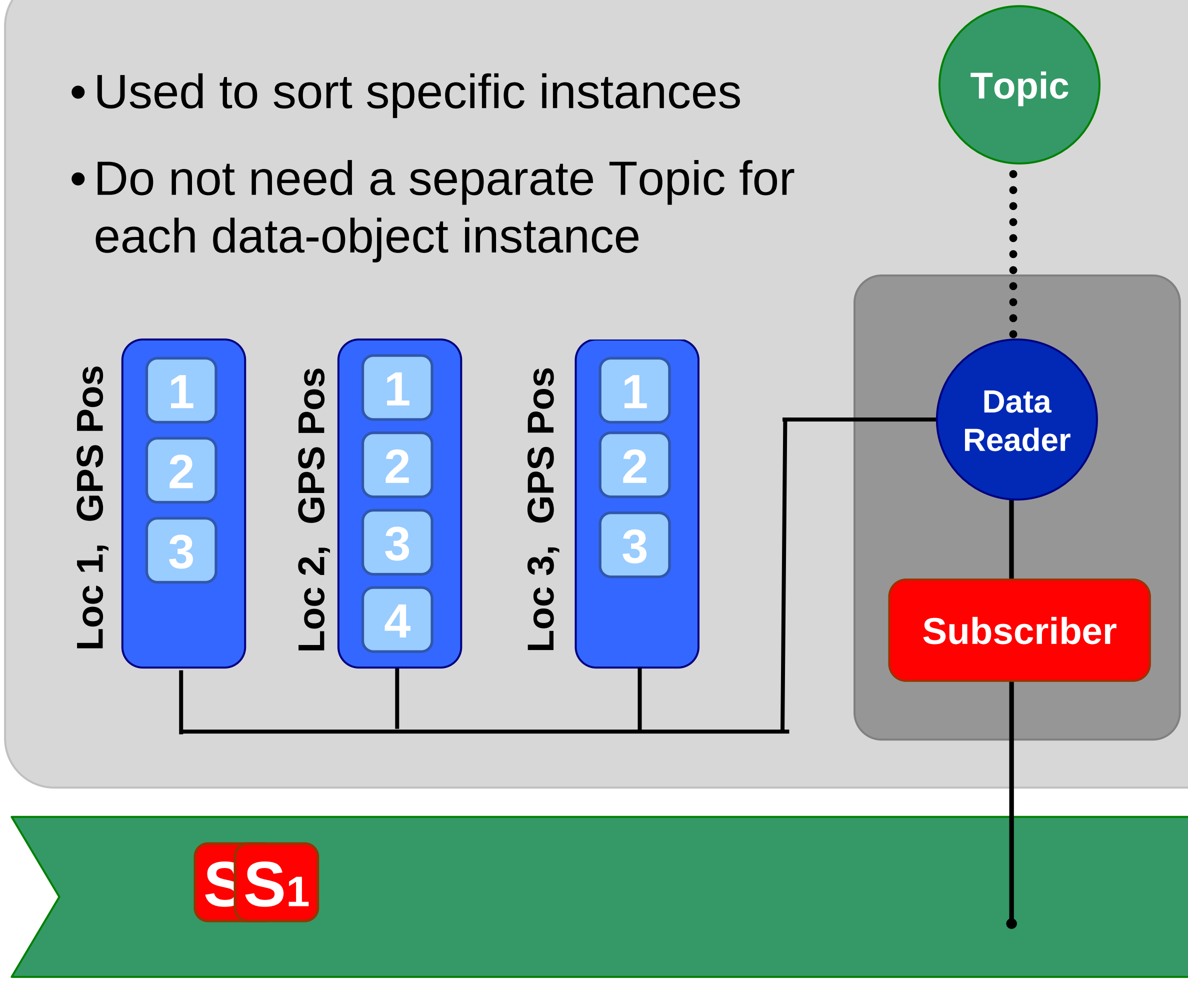


Data object addressing: Keys

Address in Global Data Space = (Topic, Key)

Multiple instances of the same topic

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

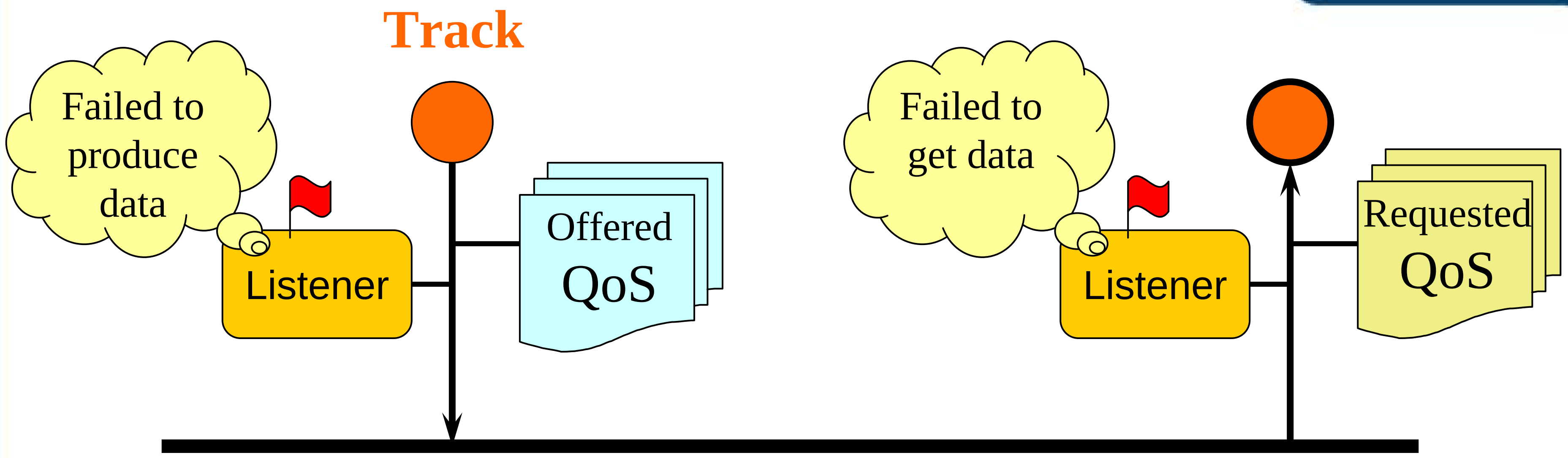


- Topic key can be any field within the Topic.

Example:

```
struct LocationInfo
{
      int LocID; //key
      GPSPos pos;
};
```

DDS communications model



Publisher declares information it has and specifies the Topic

- … and the offered QoS contract
- … and an associated listener to be alerted of any significant status changes

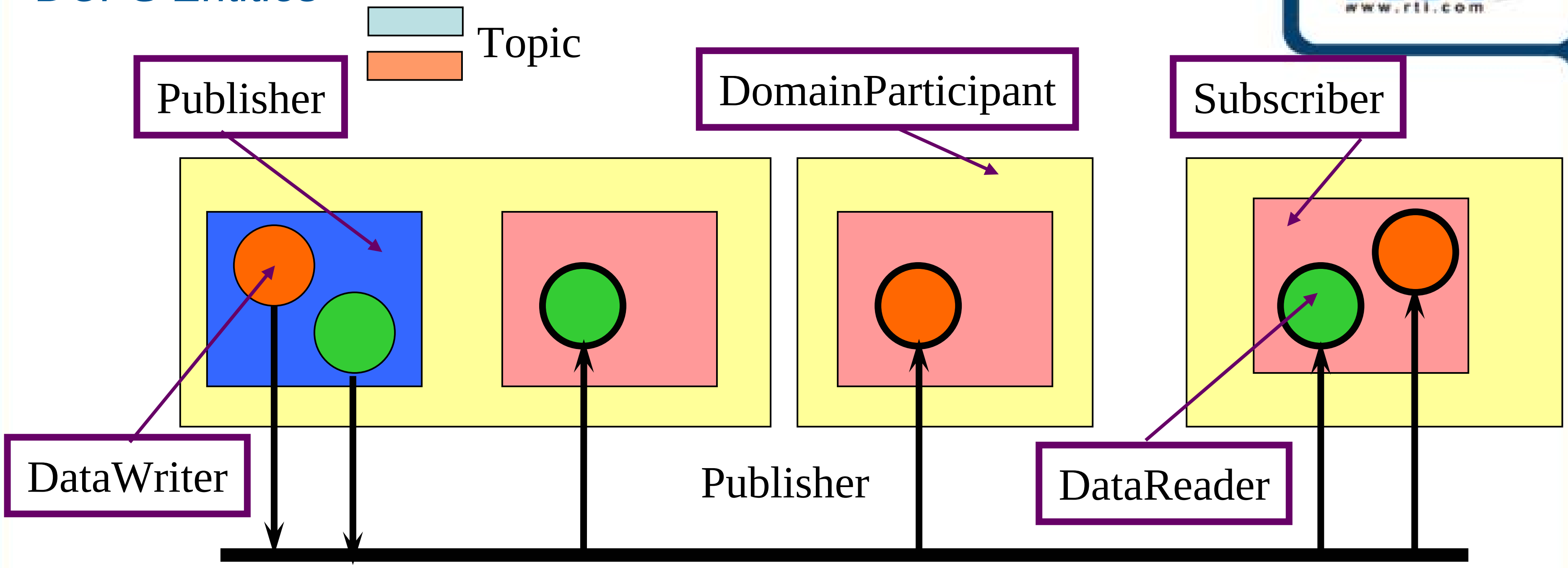
Subscriber declares information it wants and specifies the Topic

- … and the requested QoS contract
- … and an associated listener to be alerted of any significant status changes

DDS automatically discovers publishers and subscribers

- DDS ensures QoS matching and alerts of inconsistencies

DCPS Entities



DomainParticipant ~ ***Represents participation of the application in the communication collective***

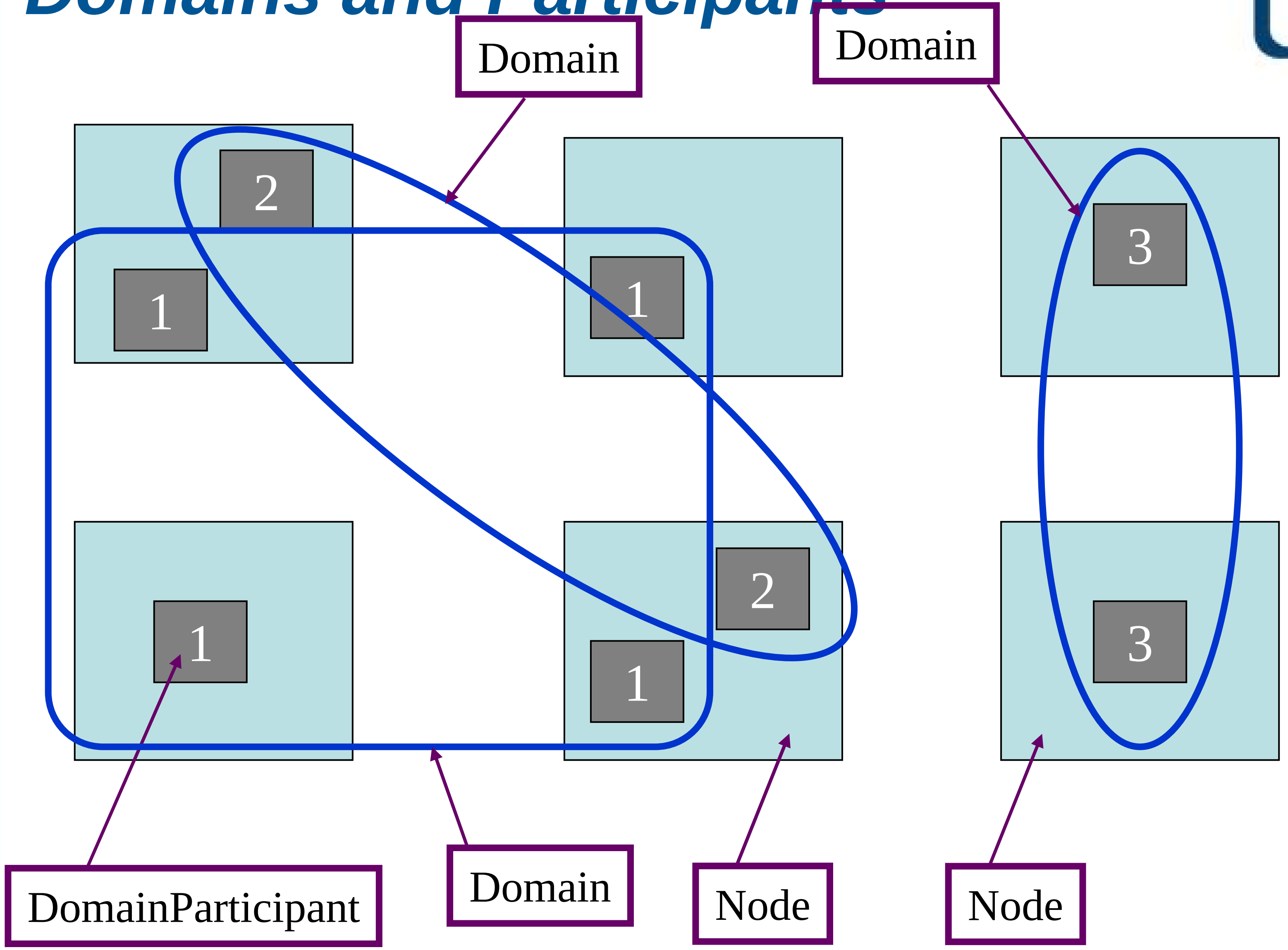
DataWriter ~ ***Accessor to write typed data on a particular*** *Topic*

Publisher ~ ***Aggregation of DataWriter objects. Responsible for disseminating information.***

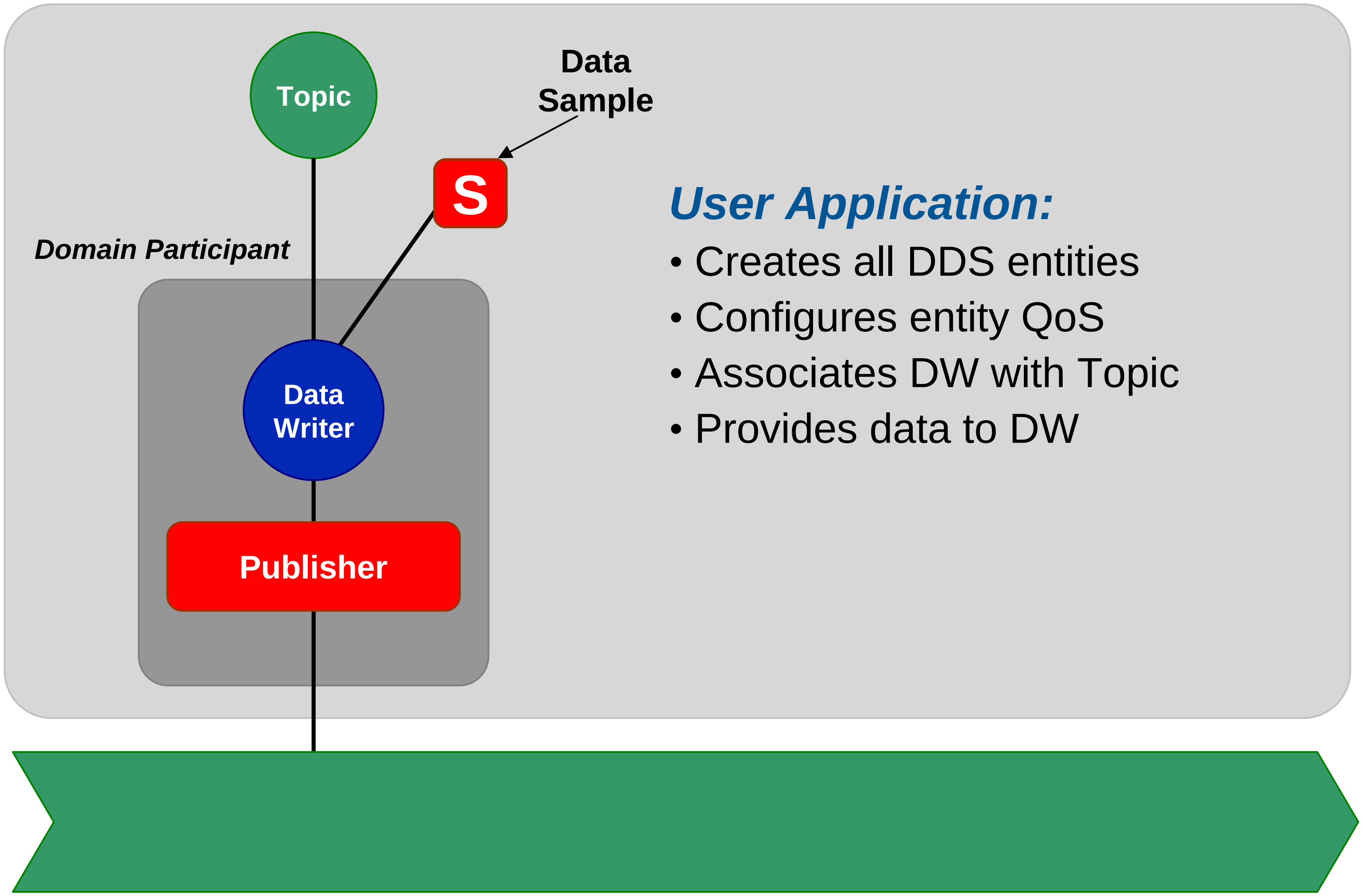
DataReader ~ ***Accessor to read typed data regarding a specific*** *Topic*

Subscriber ~ ***Aggregation of DataReader objects. Responsible for receiving information***

Domains and Participants



DDS Publication



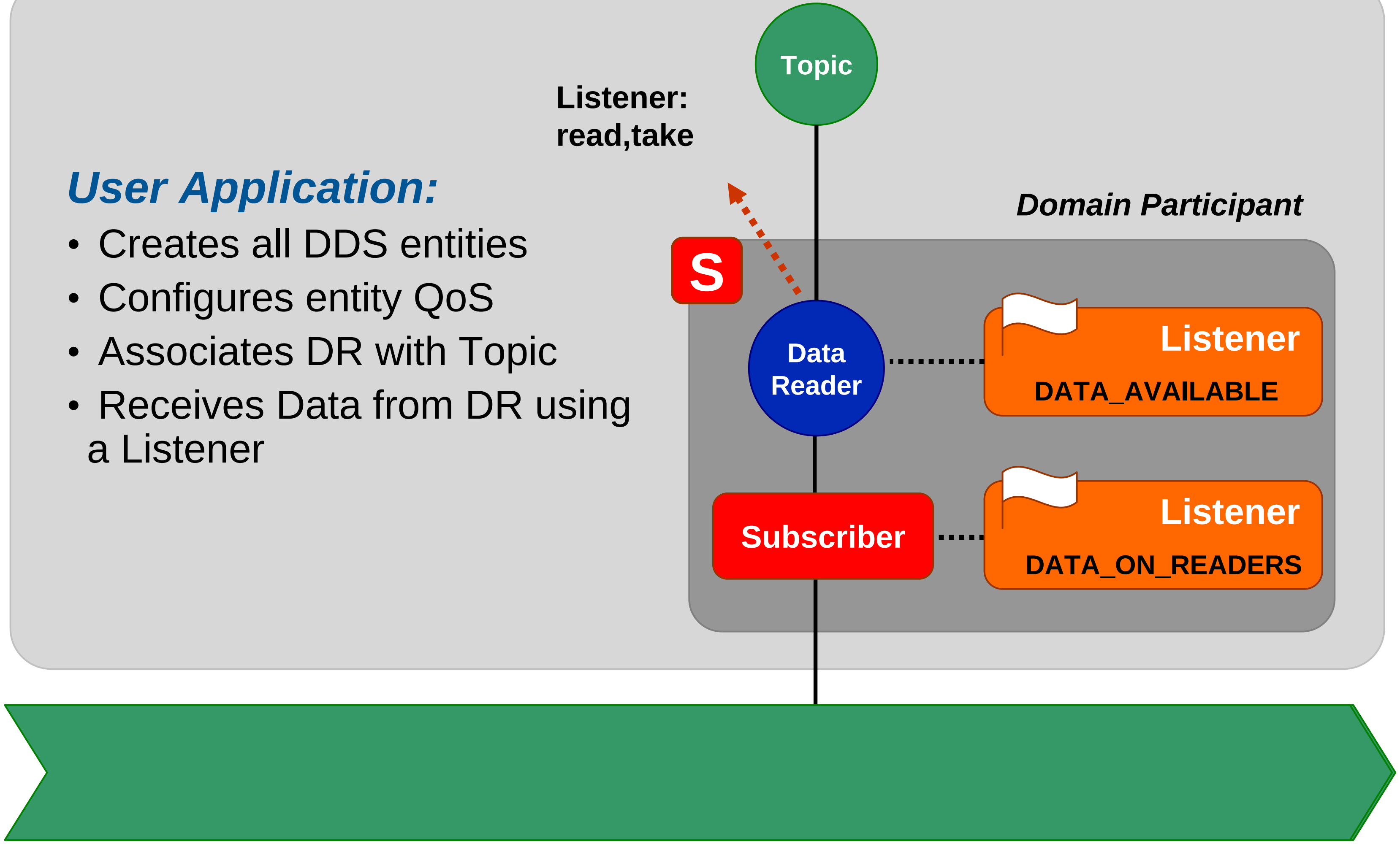
User Application:

- Creates all DDS entities
- Configures entity QoS
- Associates DW with Topic
- Provides data to DW

Example: Publication

```
Publisher publisher = domain->create_publisher(
       publisher_qos,
       publisher_listener);

Topic topic = domain->create_topic(
       “Track”, “TrackStruct”,
       topic_qos, topic_listener);

DataWriter writer = publisher->create_datawriter(
       topic, writer_qos, writer_listener);
TrackStructDataWriter twriter =
      TrackStructDataWriter::narrow(writer);

TrackStruct my_track;
twriter->write(&my_track);
```

DDS Subscription Listener



User Application:

- Creates all DDS entities
- Configures entity QoS
- Associates DR with Topic
- Receives Data from DR using a Listener

Example: Subscription

```
Subscriber  subs = domain->create_subscriber(
       subscriber_qos, subscriber_listener);

Topic topic = domain->create_topic(
       “Track”, “TrackStruct”,
       topic_qos, topic_listener);

DataReader reader = subscriber->create_datareader(
       topic, reader_qos, reader_listener);

// Use listener-based or wait-based access
```

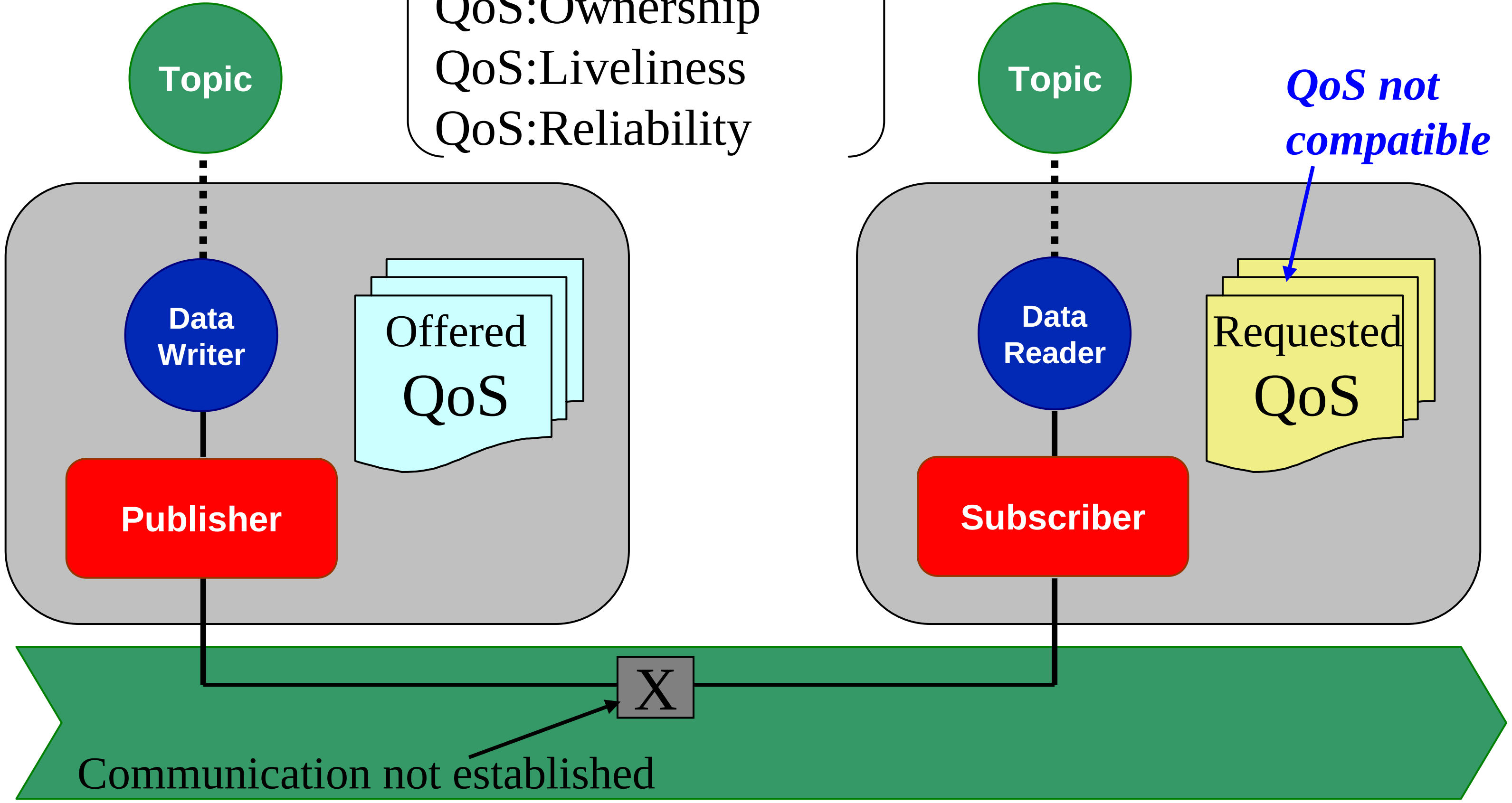
How to get data (listener-based)

```
Listener listener = new MyListener();
reader->set_listener(listener);

MyListener::on_data_available( DataReader reader )
{
    TrackStructSeq received_data;
    SampleInfoSeq sample_info;
    TrackStructDataReader treader =
        TrackStructDataReader::narrow(reader);

    treader->take( &received_data,
                   &sample_info, …)

    // Use received_data
}
```

QoS Contract "Request / Offered"

QoS:Durability
QoS:Presentation
QoS:Deadline
QoS:Latency_Budget
QoS:Ownership
QoS:Liveliness
QoS:Reliability

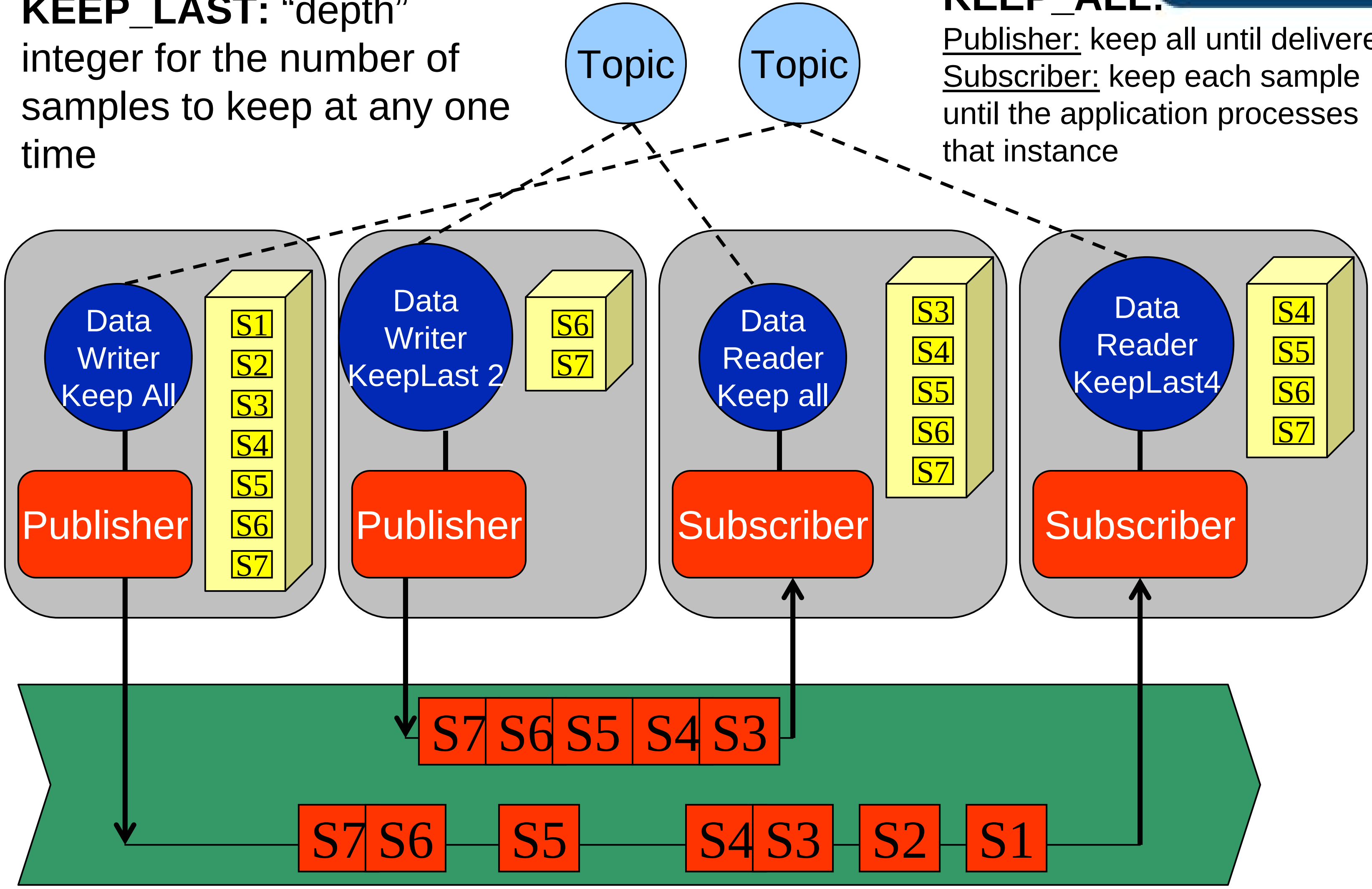
QoS Request / Offered:
Ensure that the compatible QoS parameters are set.



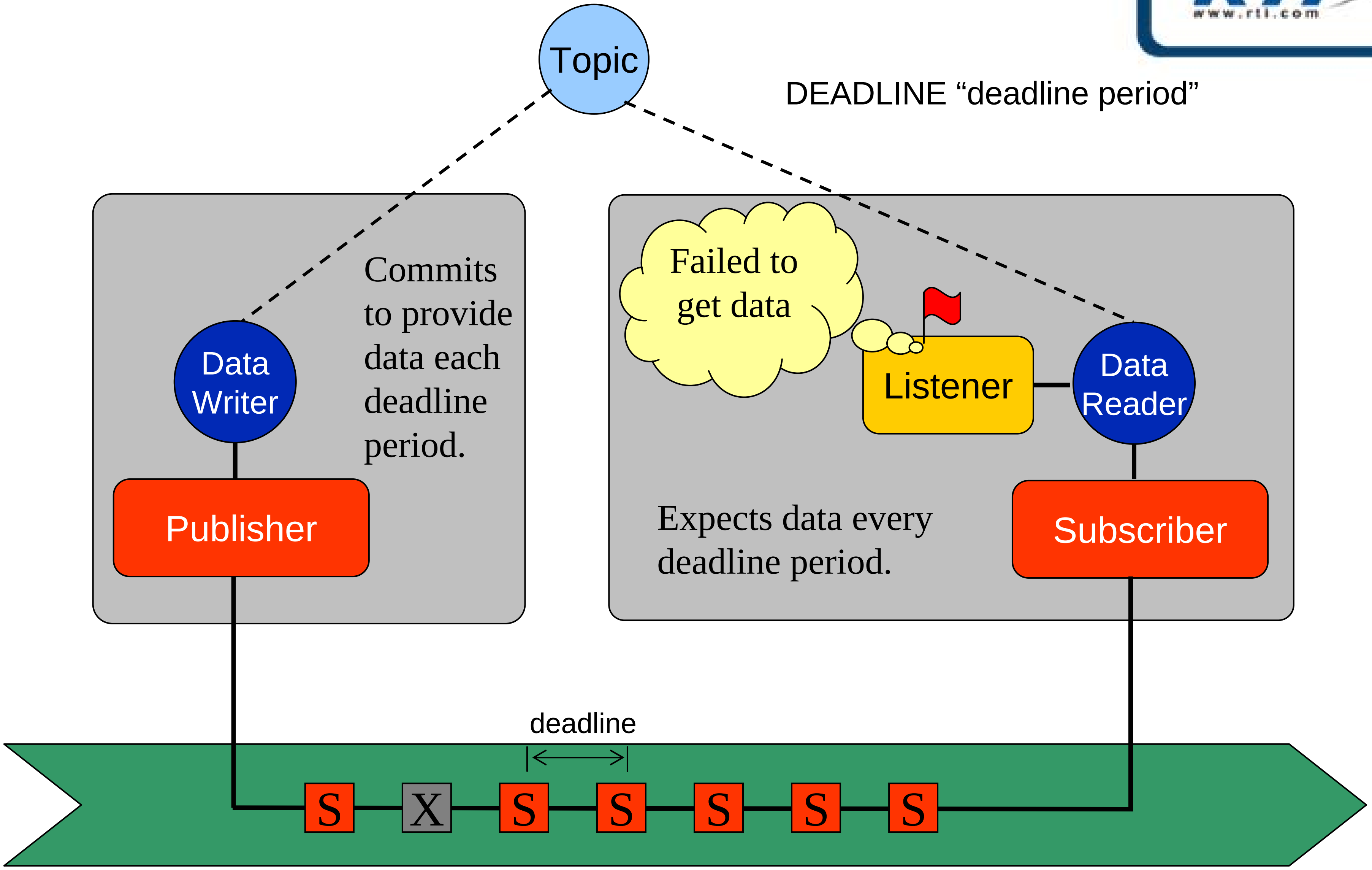
QoS: History: Last x or All

KEEP_LAST: "depth" integer for the number of samples to keep at any one time

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

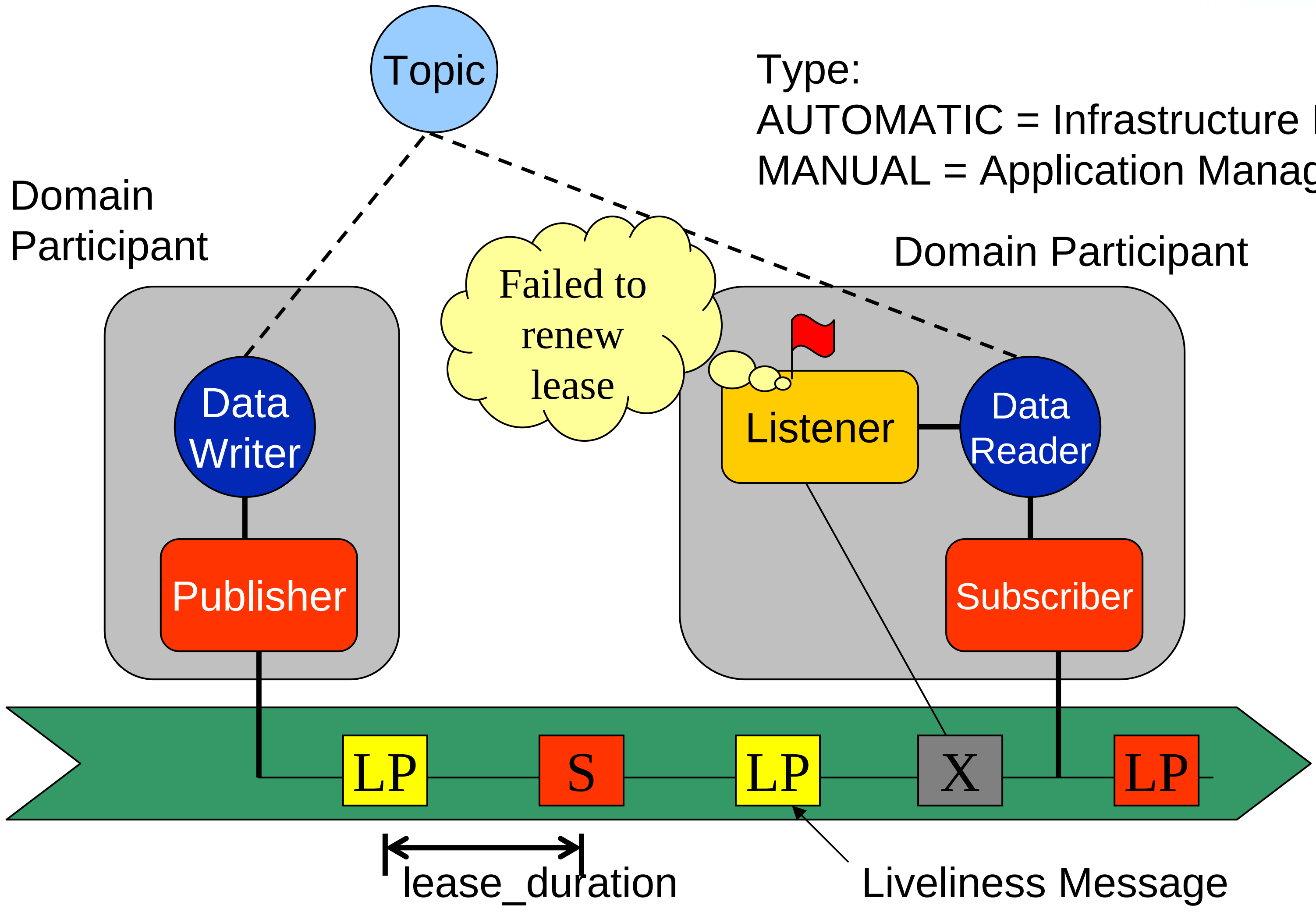


QoS: Deadline


Topic
DEADLINE "deadline period"
Data Writer
Publisher
Commits to provide data each deadline period.
Failed to get data
Listener
Data Reader
Subscriber
Expects data every deadline period.
deadline
S
X
S
S
S
S
S

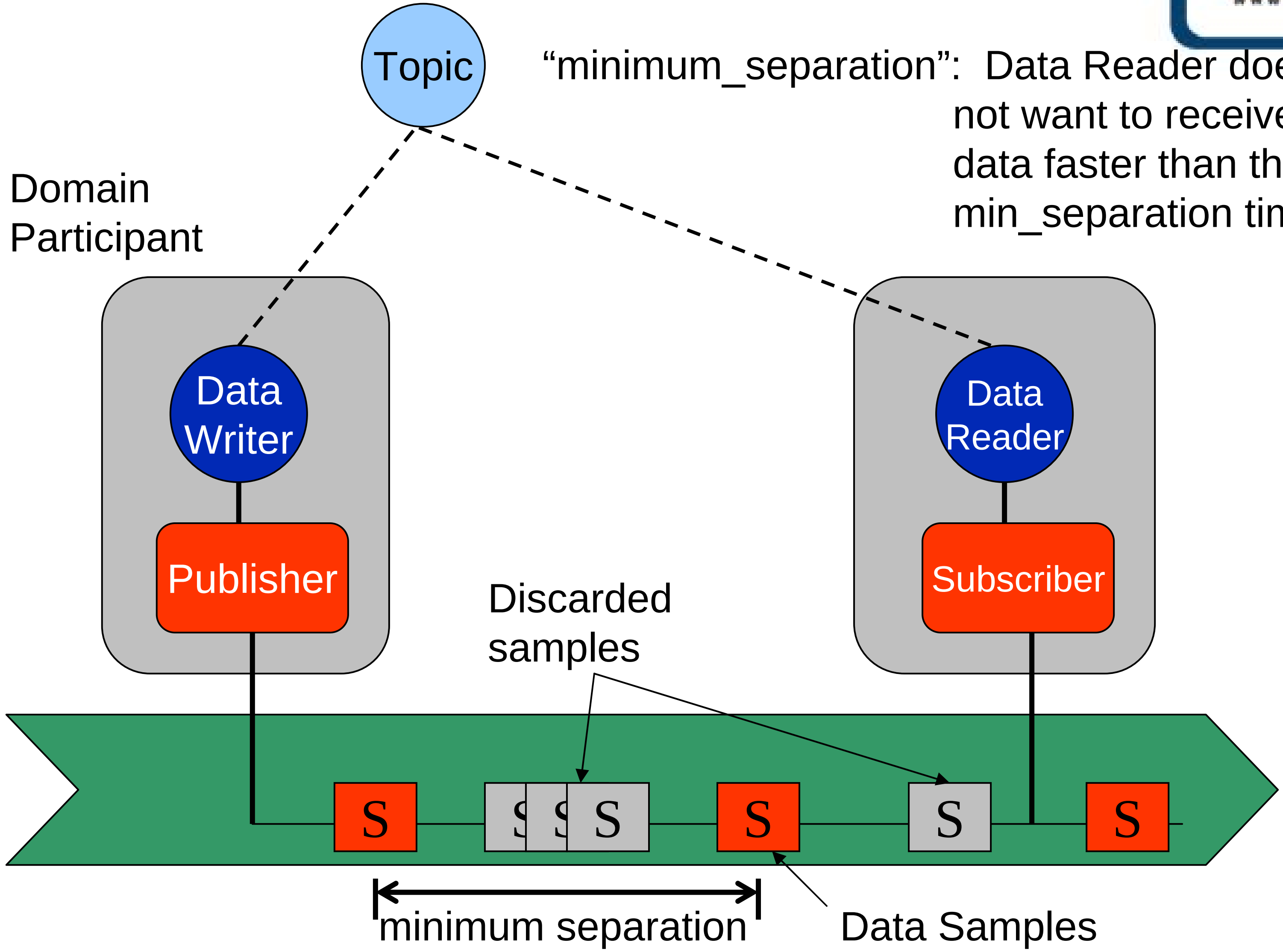
QoS: Liveliness – Type, Duration

Type:
AUTOMATIC = Infrastructure Managed
MANUAL = Application Managed



QoS: Time_Based_Filter

“minimum_separation”: Data Reader does not want to receive data faster than the min_separation time



QoS: Quality of Service (1/2)

QoS Policy	Concerns	RxO	Changeable
DEADLINE	T,DR,DW	YES	YES
LATENCY BUDGET	T,DR,DW	YES	YES
READER DATA LIFECYCLE	DR	N/A	YES
WRITER DATA LIFECYCLE	DW	N/A	YES
TRANSPORT PRIORITY	T,DW	N/A	YES
LIFESPAN	T,DW	N/A	YES
LIVELINESS	T,DR,DW	YES	NO
TIME BASED FILTER	DR	N/A	YES
RELIABILITY	T,DR,DW	YES	NO
DESTINATION ORDER	T,DR	NO	NO

QoS: Quality of Service (2/2)

QoS Policy	Concerns	RxO	Changeable
USER DATA	DP,DR,DW	NO	YES
TOPIC DATA	T	NO	YES
GROUP DATA	P,S	NO	YES
ENTITY FACTORY	DP, P, S	NO	YES
PRESENTATION	P,S	YES	NO
OWNERSHIP	T	YES	NO
OWNERSHIP STRENGTH	DW	N/A	YES
PARTITION	P,S	NO	YES
DURABILITY	T,DR,DW	YES	NO
HISTORY	T,DR,DW	NO	NO
RESOURCE LIMITS	T,DR,DW	NO	NO

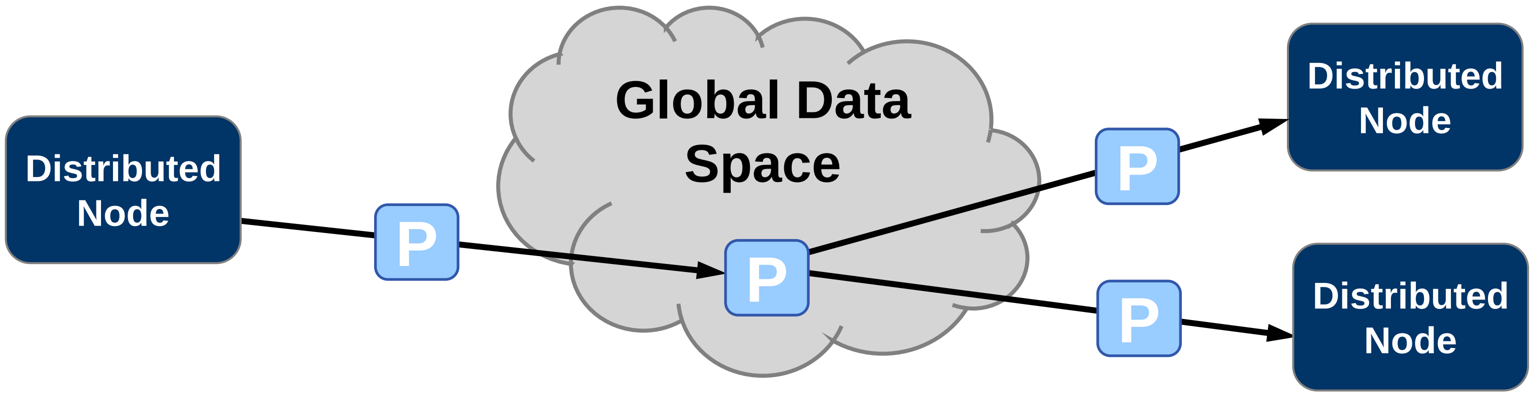
Summary

DDS targets applications that need to distribute data in a real-time environment

DDS is highly configurable by QoS settings

DDS provides a shared "global data space"

- Any application can publish data it has
- Any application can subscribe to data it needs
- Automatic discovery
- Facilities for fault tolerance
- Heterogeneous systems easily accommodated



Thank you

References:

OMG DDS specification:

http://www.omg.org/cgi-bin/doc?ptc/04-04-12

General material on DDS and RTI's implementation:

http://www.rti.com/dds

Comments/questions: gerardo@rti.com