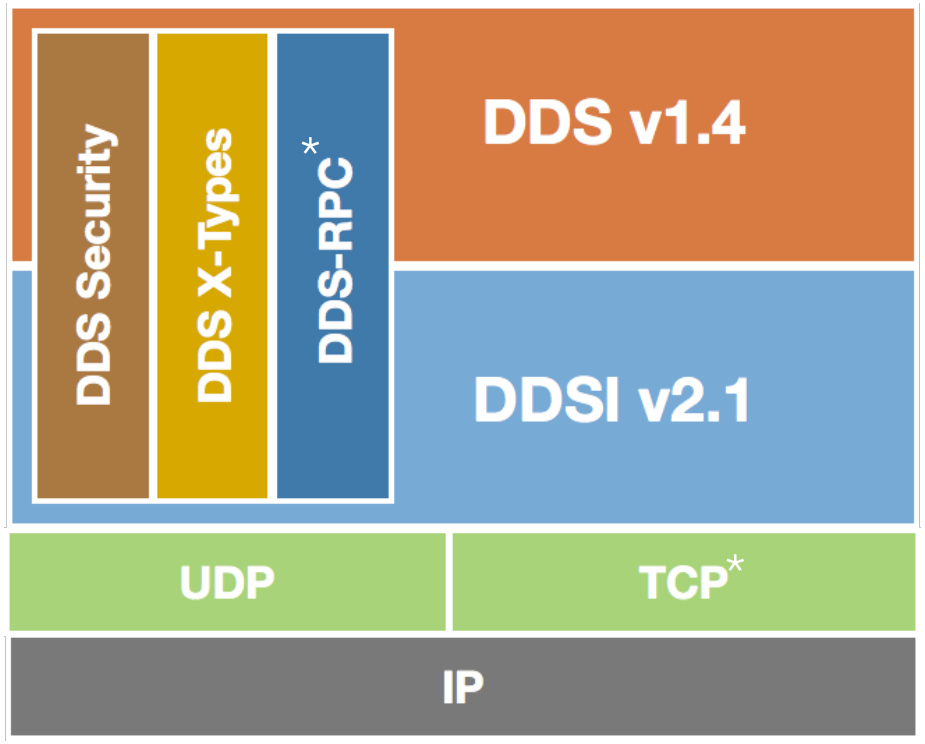
DDS

The IoT Data Sharing Standard

The DDS Standard

DDS
The IoT Data Sharing Standard

- Introduced in 2004, DDS is an Object Management Group (OMG) Standard for efficient, secure and interoperable, platform- and programming-language independent data sharing
- DDS standardises:
  - Programming API
  - Interoperable wire-protocol
  - Extensible Type System
  - Data Modeling
  - Security Framework
  - Remote Procedure Call

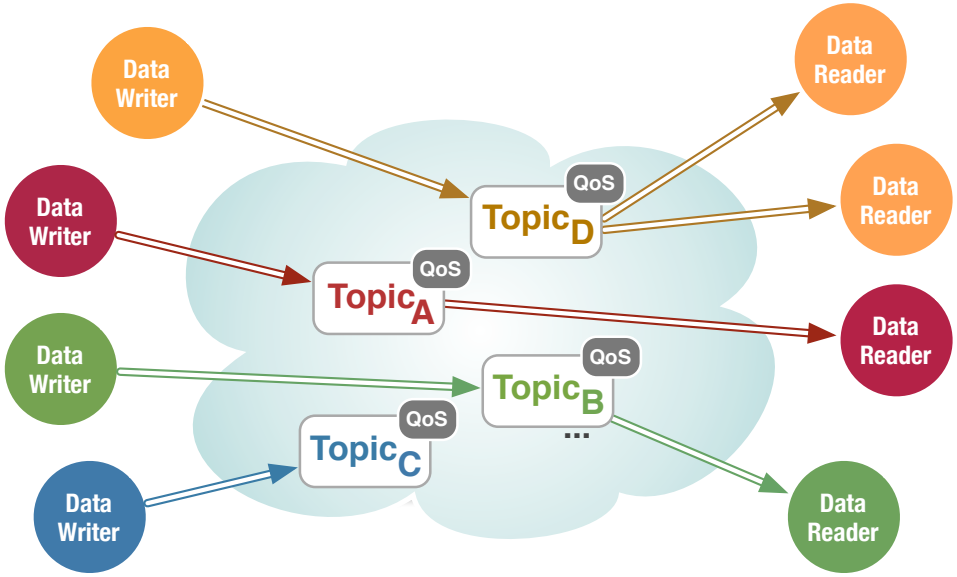


(*) Under Finalisation

How is DDS Different/Better?

Higher Level Abstraction

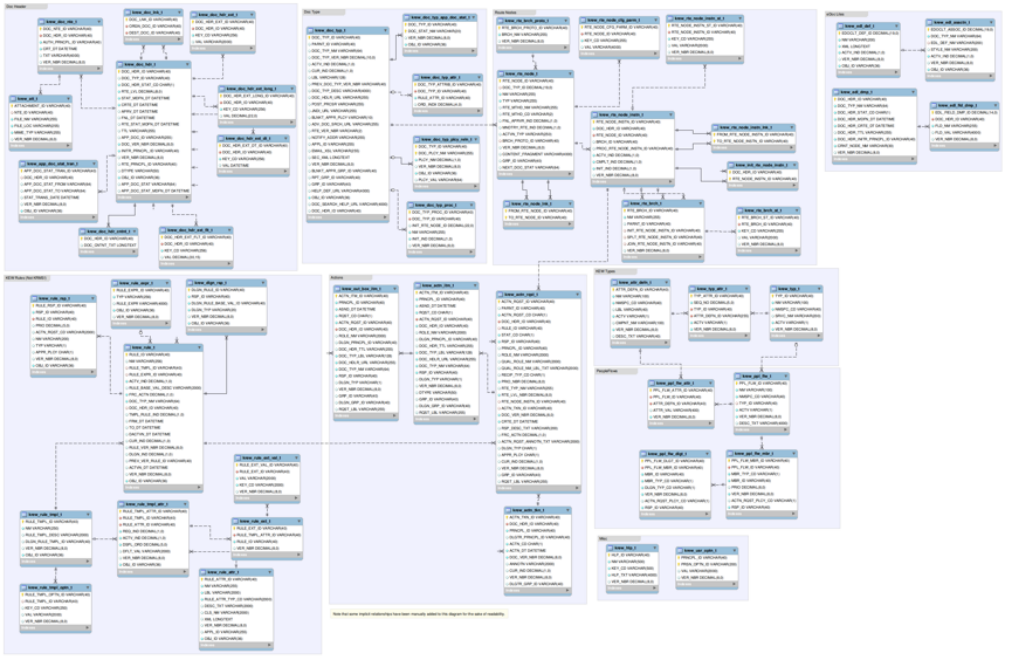
- Provides a **Distributed Data Space** abstraction where **applications** can **autonomously** and **asynchronously read** and **write data**
- Its built-in **dynamic discovery isolates applications** from **network topology** and **connectivity details**



DDS Global Data Space

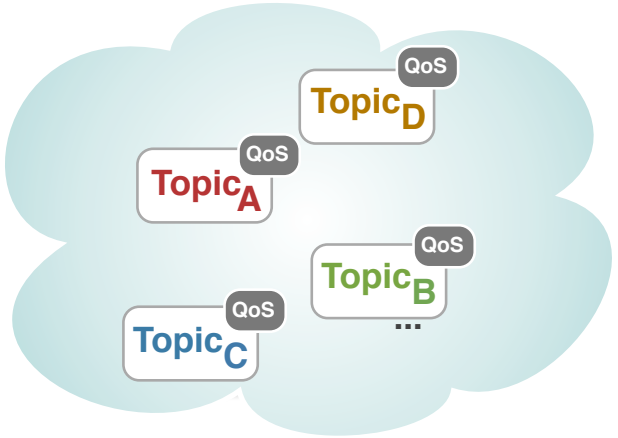
Data Centricity

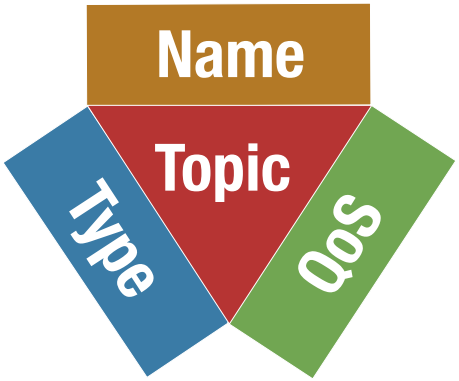
- DDS supports the definition of **Common Information Models**. These data models define the system's Lingua Franca
- DDS types are extensible and evolvable, thus allowing incremental updates and upgrades



Topic

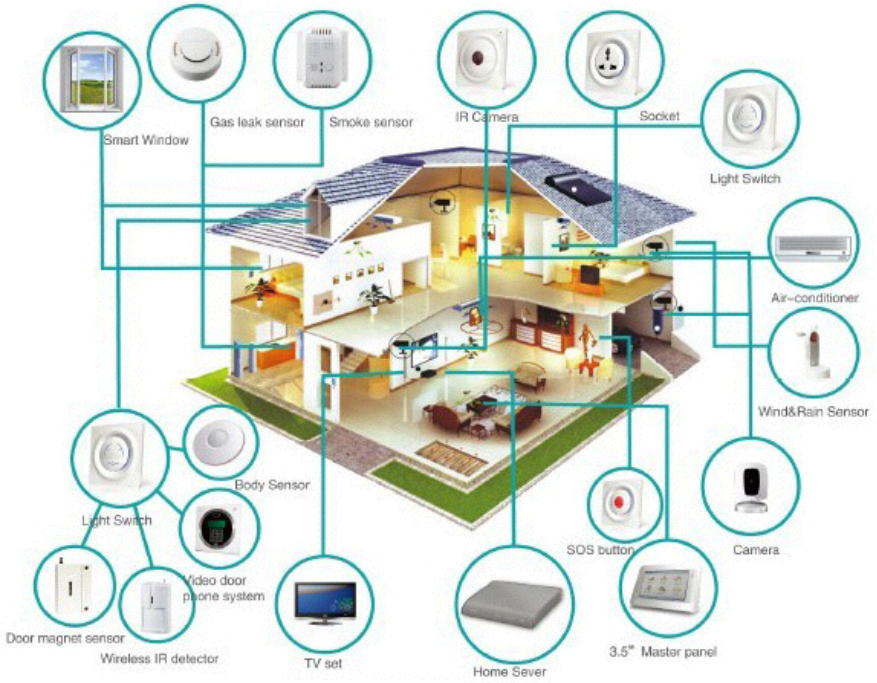
- A Topic defines a **domain-wide** information's class
- A **Topic** is defined by means of a (name, type, qos) tuple, where
  - **name:** identifies the topic within the domain
  - **type:** is the programming language type associated with the topic. Types are extensible and evolvable
  - **qos**: is a collection of policies that express the non-functional properties of this topic, e.g. reliability, persistence, etc.





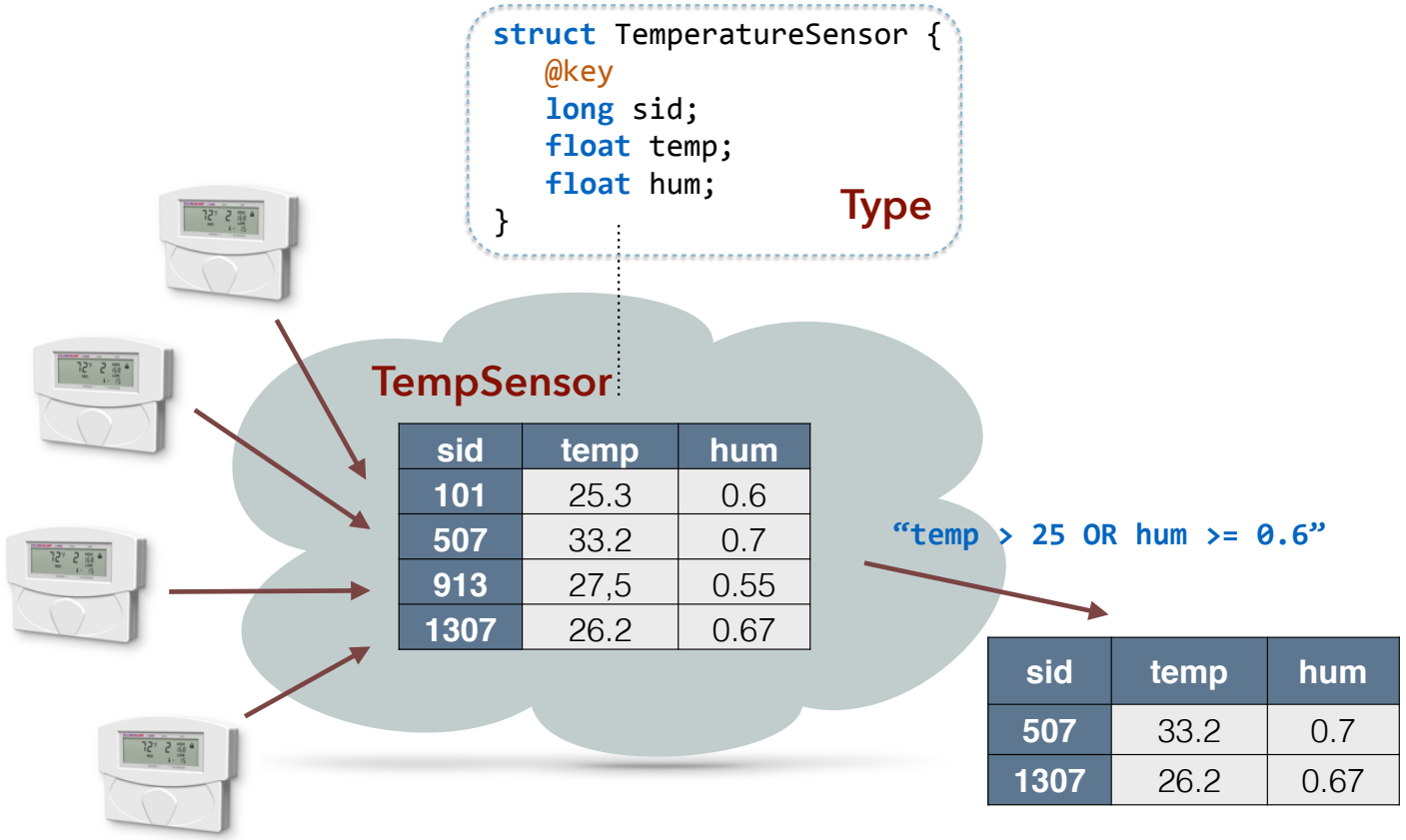
Topic and Instances

- As explained in the previous slide **a topic defines a class/type of information**
- Topics can be defined as **Singleton** or can have multiple **Instances**
- **Topic Instances** are identified by means of the topic key
- A Topic **Key** is identified by a **tuple of attributes** -- like in databases
- Remarks:
  - A **Singleton** topic has a single domain-wide instance
  - A “regular” Topic can have as many instances as the number of different key values, *e.g., if the key is an 8-bit character then the topic can have 256 different instances*



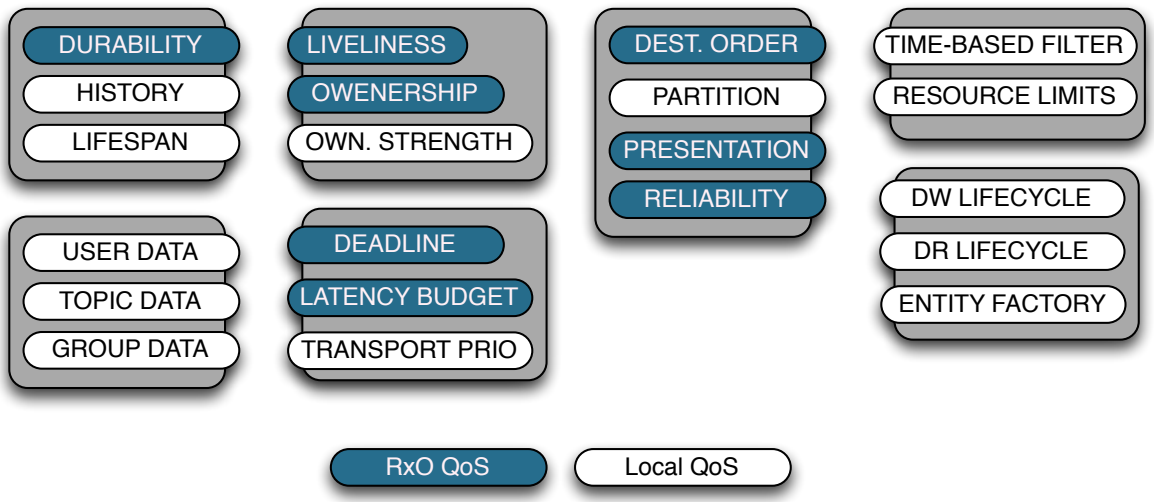
Content Awareness

- DDS "**knows**" about application data types and uses this information provide type-safety and content-based routing



QoS Policies

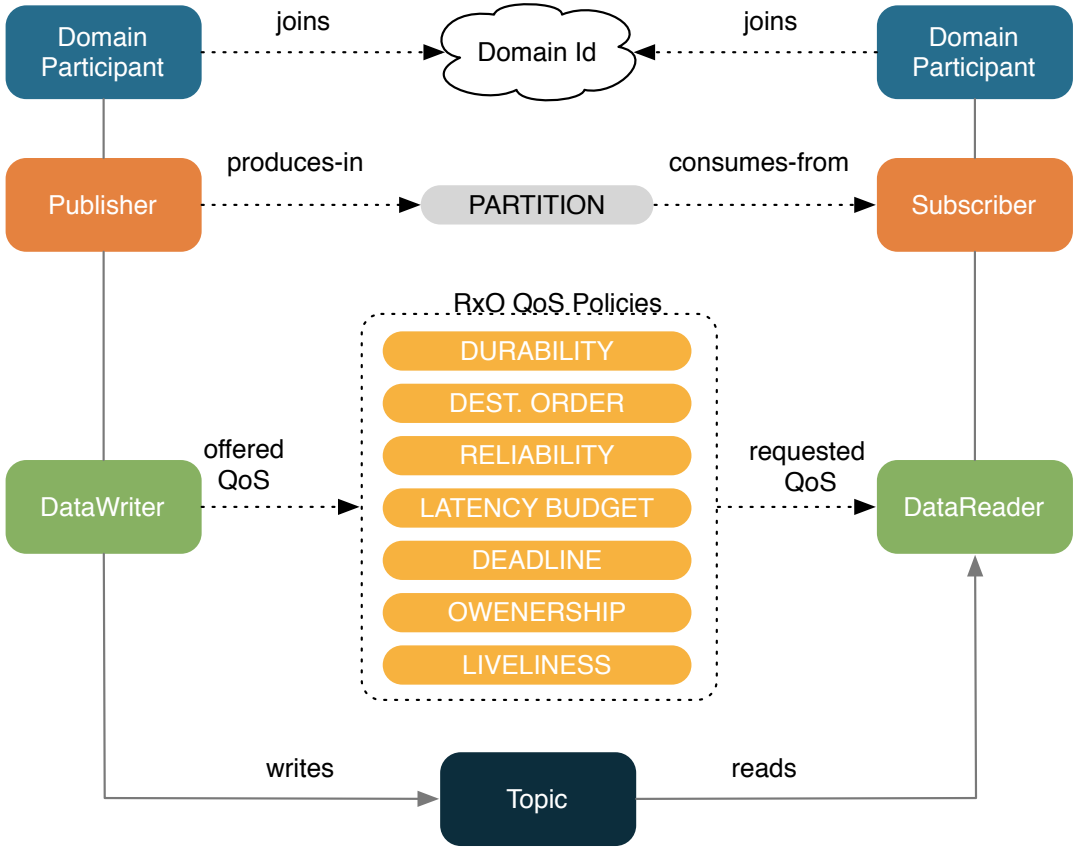
- DDS provides a rich set of QoS-Policies to control local as well as end-to-end properties of data sharing
- Some QoS-Policies are matched based on a Request vs. Offered (RxO) Model



QoS Model

For data to flow from a DataWriter (DW) to one or many DataReader (DR) a few conditions have to apply:

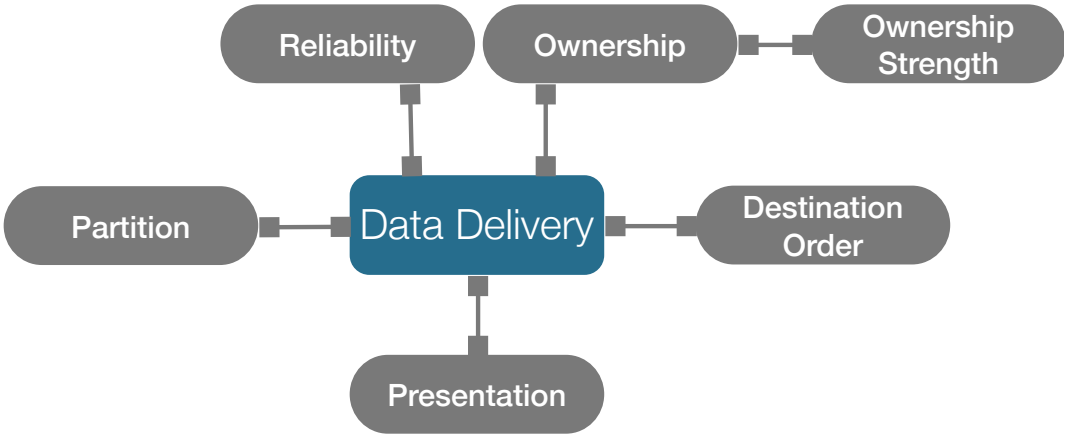
- The DR and DW domain participants have to be in the same domain
- The partition expression of the DR's Subscriber and the DW's Publisher should match (in terms of regular expression match)
- The QoS Policies offered by the DW should exceed or match those requested by the DR



Data Delivery

Data Delivery QoS Policies provide control over:

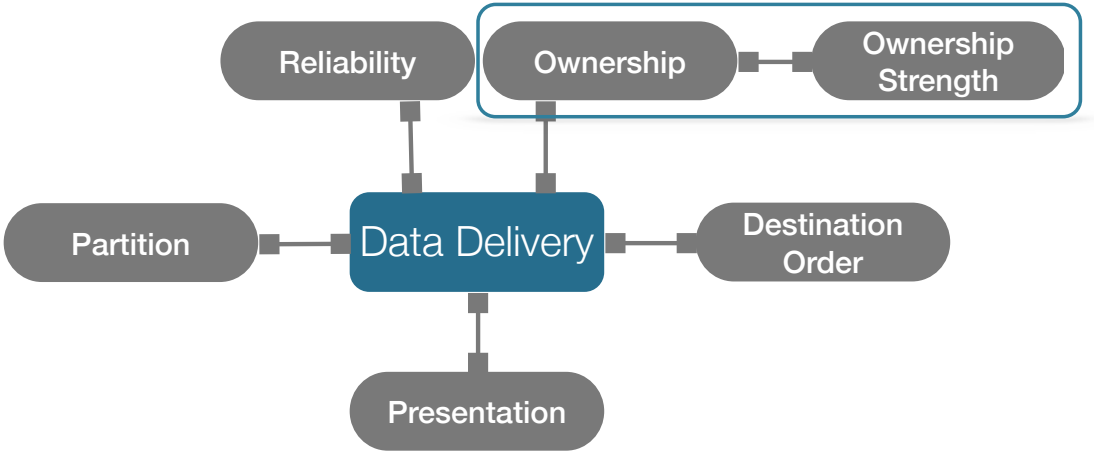
- **who** delivers data
- **where** data is delivered, and
- **how** data is delivered



Data Delivery

Data Delivery QoS Policies provide control over:

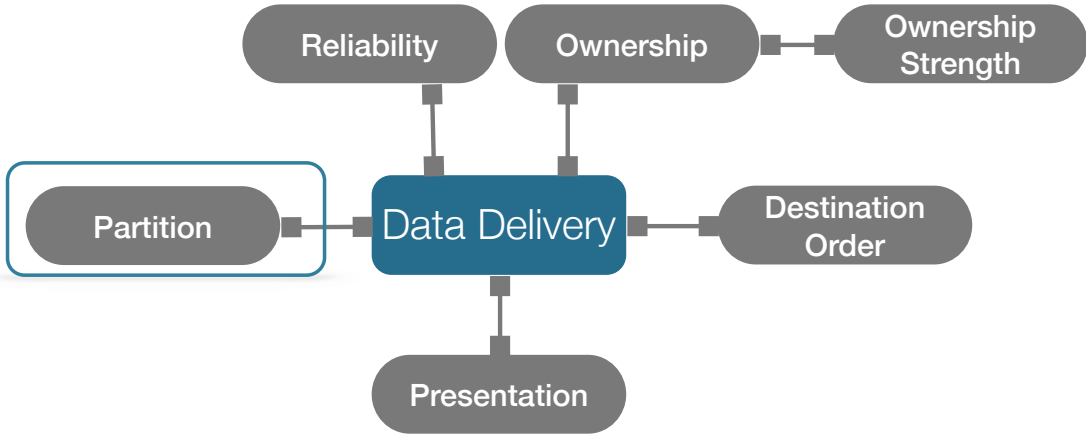
- **who** delivers data
- **where** data is delivered, and
- **how** data is delivered



Data Delivery

Data Delivery QoS Policies provide control over:

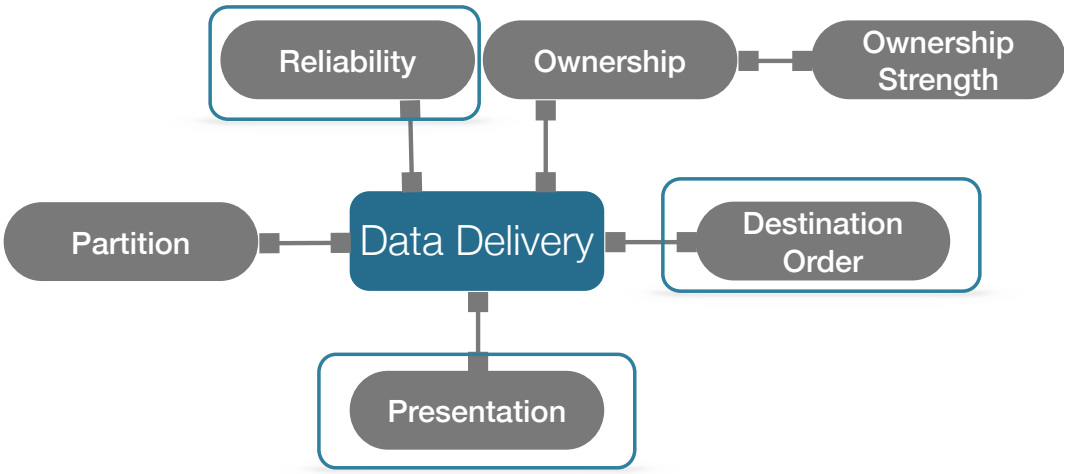
- **who** delivers data
- **where** data is delivered, and
- **how** data is delivered



Data Delivery

Data Delivery QoS Policies provide control over:

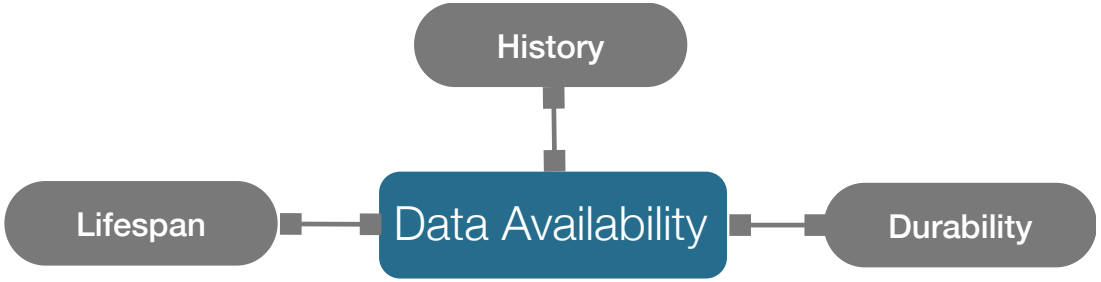
- **who** delivers data
- **where** data is delivered, and
- **how** data is delivered



Data Availability

Data Availability QoS Policies provide control over data availability with respect to:

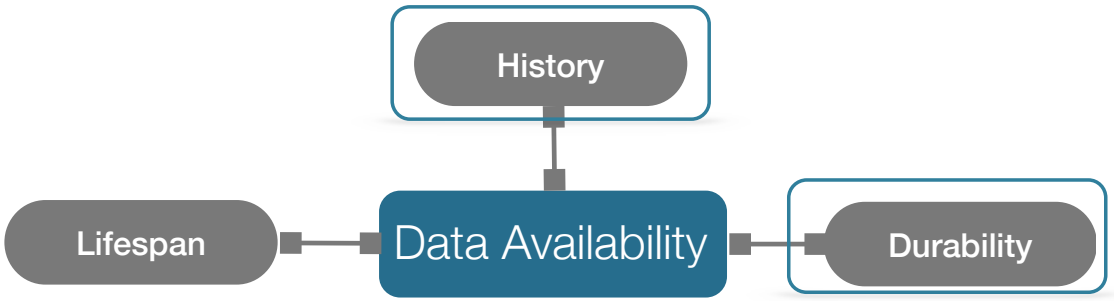
- Temporal Decoupling (late Joiners)
- Temporal Validity



Data Availability

Data Availability QoS Policies provide control over data availability with respect to:

- Temporal Decoupling (late Joiners)
- Temporal Validity



Data Availability

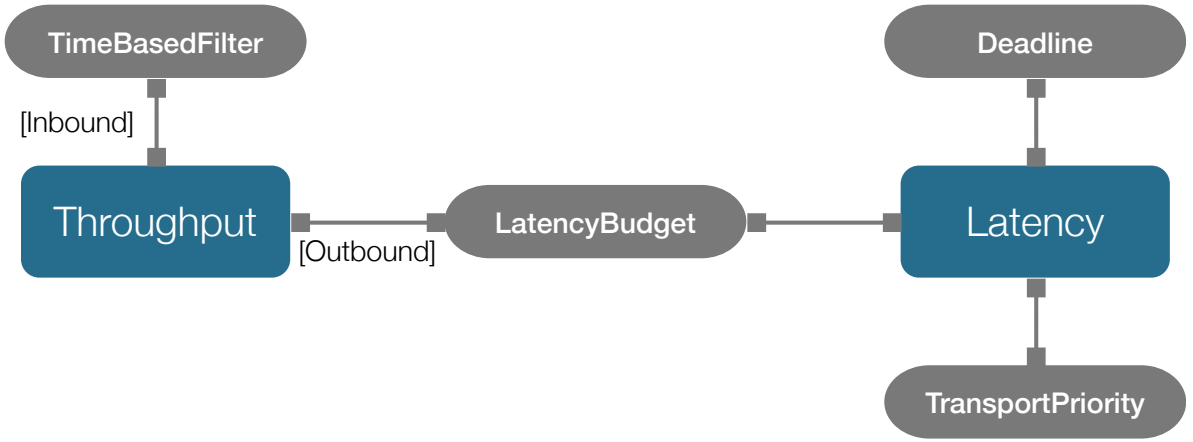
Data Availability QoS Policies provide control over data availability with respect to:

- Temporal Decoupling (late Joiners)
- Temporal Validity

Temporal Properties

Several policies provide control over temporal properties, specifically:

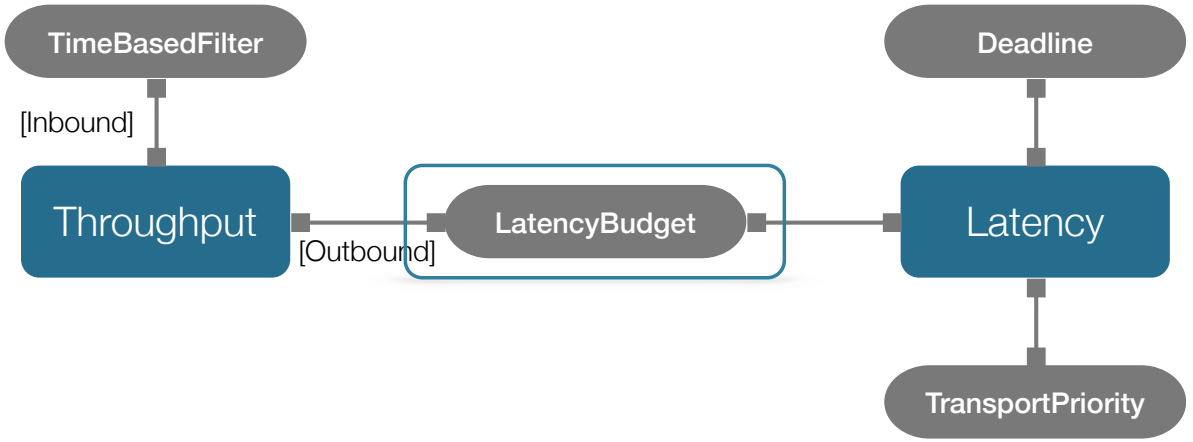
- Outbound Throughput
- Inbound Throughput
- Latency



Temporal Properties

Several policies provide control over temporal properties, specifically:

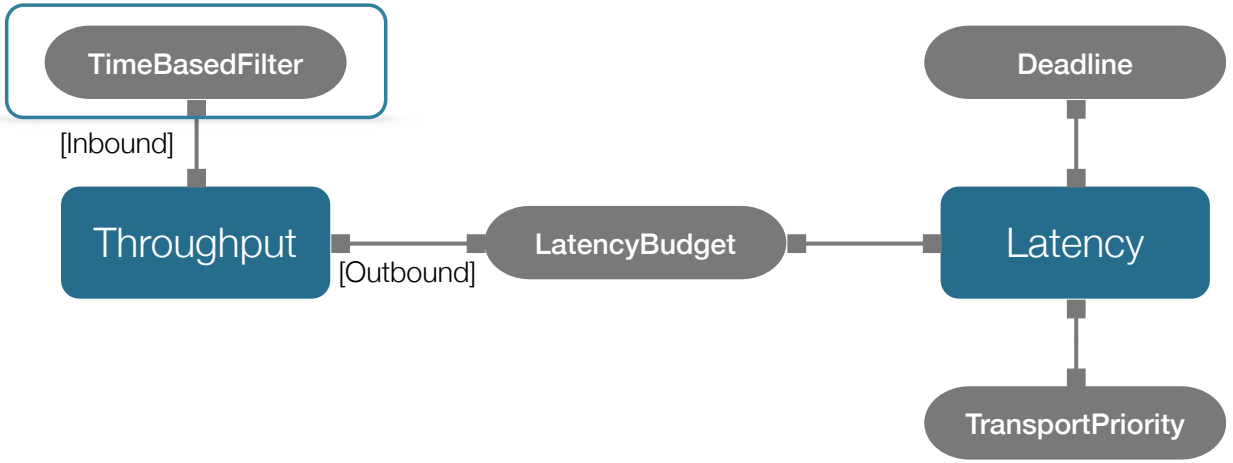
- Outbound Throughput
- Inbound Throughput
- Latency



Temporal Properties

Several policies provide control over temporal properties, specifically:

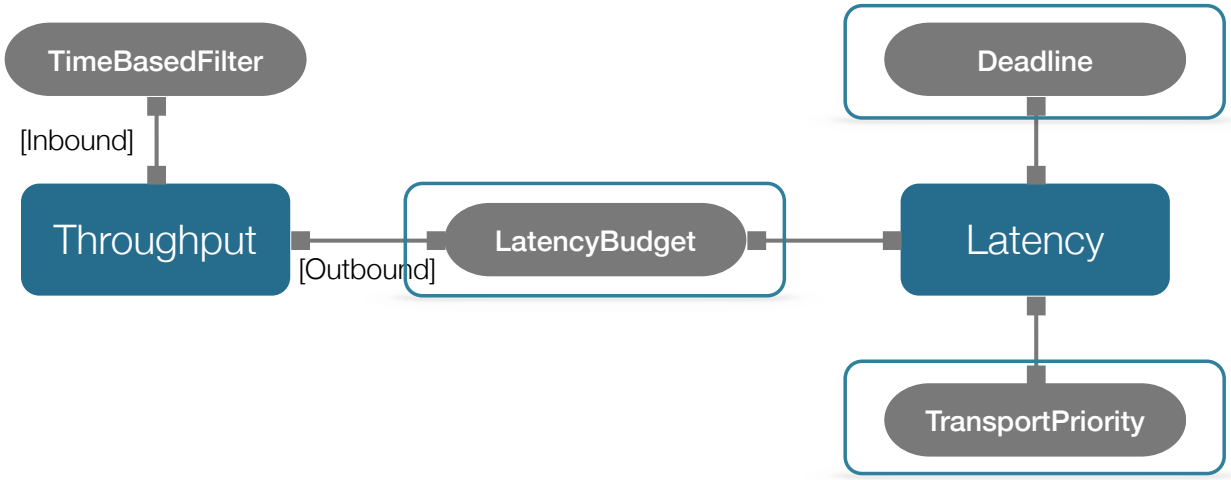
- Outbound Throughput
- Inbound Throughput
- Latency



Temporal Properties

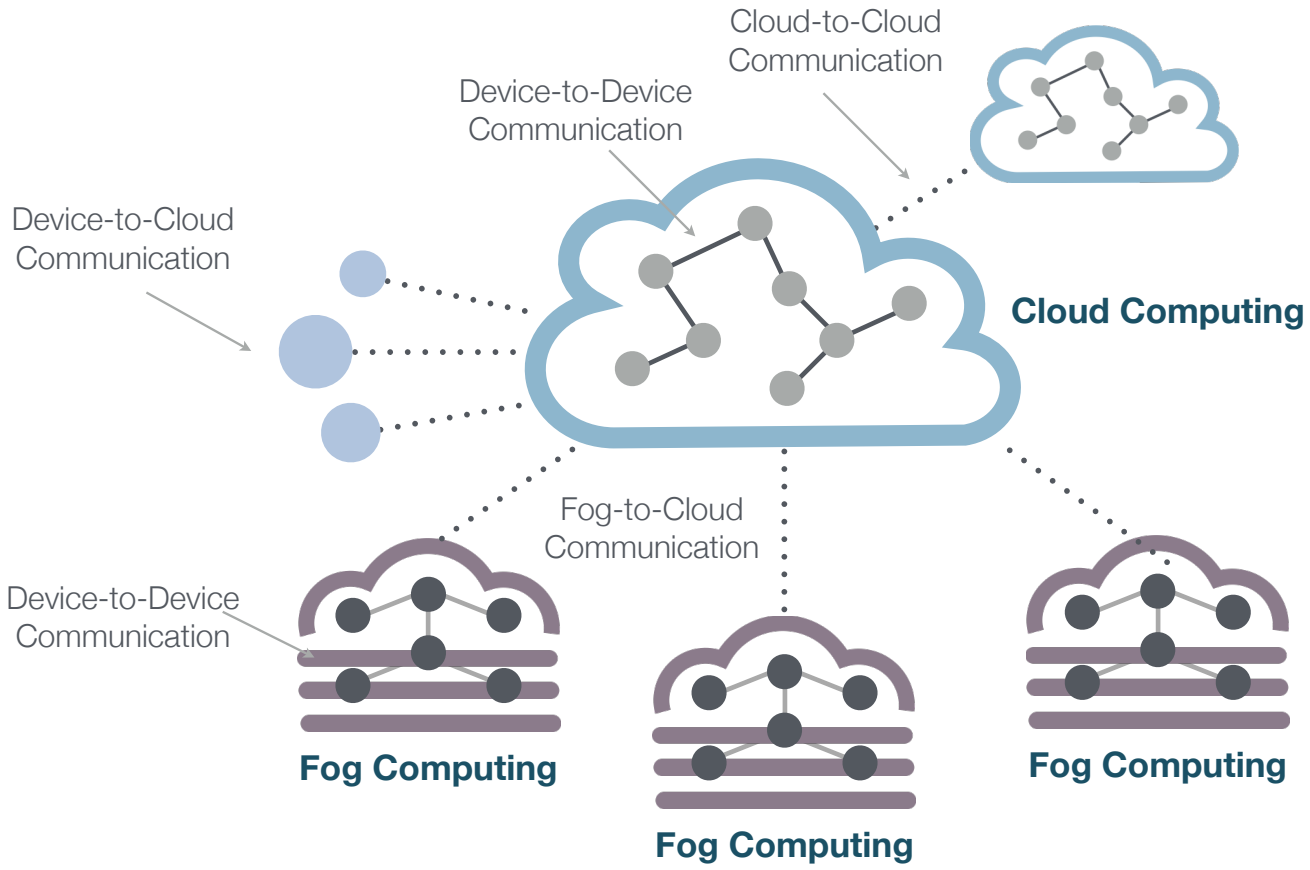
Several policies provide control over temporal properties, specifically:

- Outbound Throughput
- Inbound Throughput
- Latency



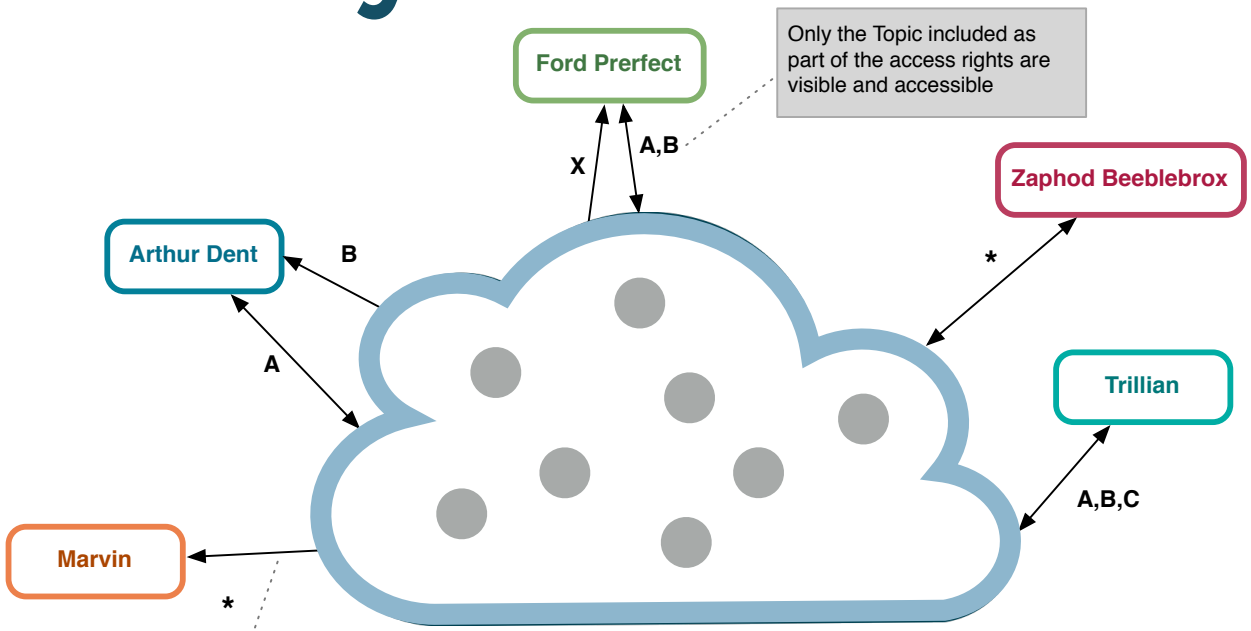
Cloud and Fog/Edge Computing

- DDS supports both the **Cloud** and the **Fog Computing** Paradigm
- DDS natively supports:
  - Device-to-Device Communication
  - Device-to-Cloud Communication



Security

- Support for fine grained access control
- Support for Symmetric and Asymmetric Authentication
- Standard Authentication, Access Control, Crypto, and Logging plug-in API



Identity	Access Rights
Arthur Dent	A(r,w), B(r)
Ford Prerfect	A(r,w), B(r,w), X(r)
Zaphod Beeblebrox	*(r,w)
Trillian	A(r,w), B(r,w), C(r,w)
Marvin	*(r)

Platform Independent

- DDS is independent from the
  - Programming language,
  - Operating System
  - HW architecture