Mastering DDS QoS

Angelo Corsaro, PhD

CTO, ADLINK Tech. Inc.
Co-Chair, OMG DDS-SIG
angelo.corsaro@adlinktech.com

PRISMTECH™
AN ADLINK COMPANY

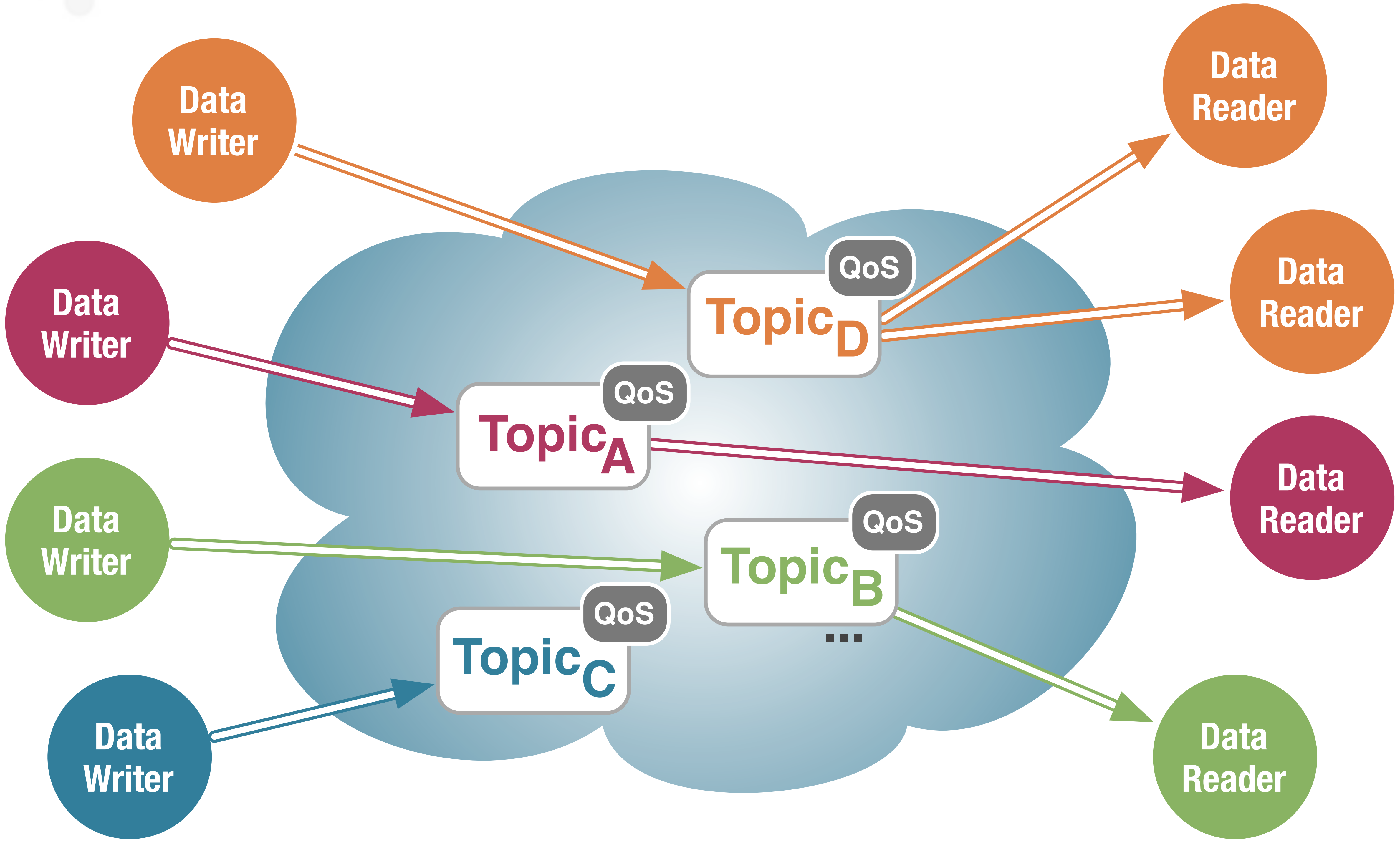
Copyright PrismTech, 2017

DDS Refresher

Copyright PrismTech, 2017

DDS Abstraction

DDS provides applications with a Virtual **Global Data Space** abstraction

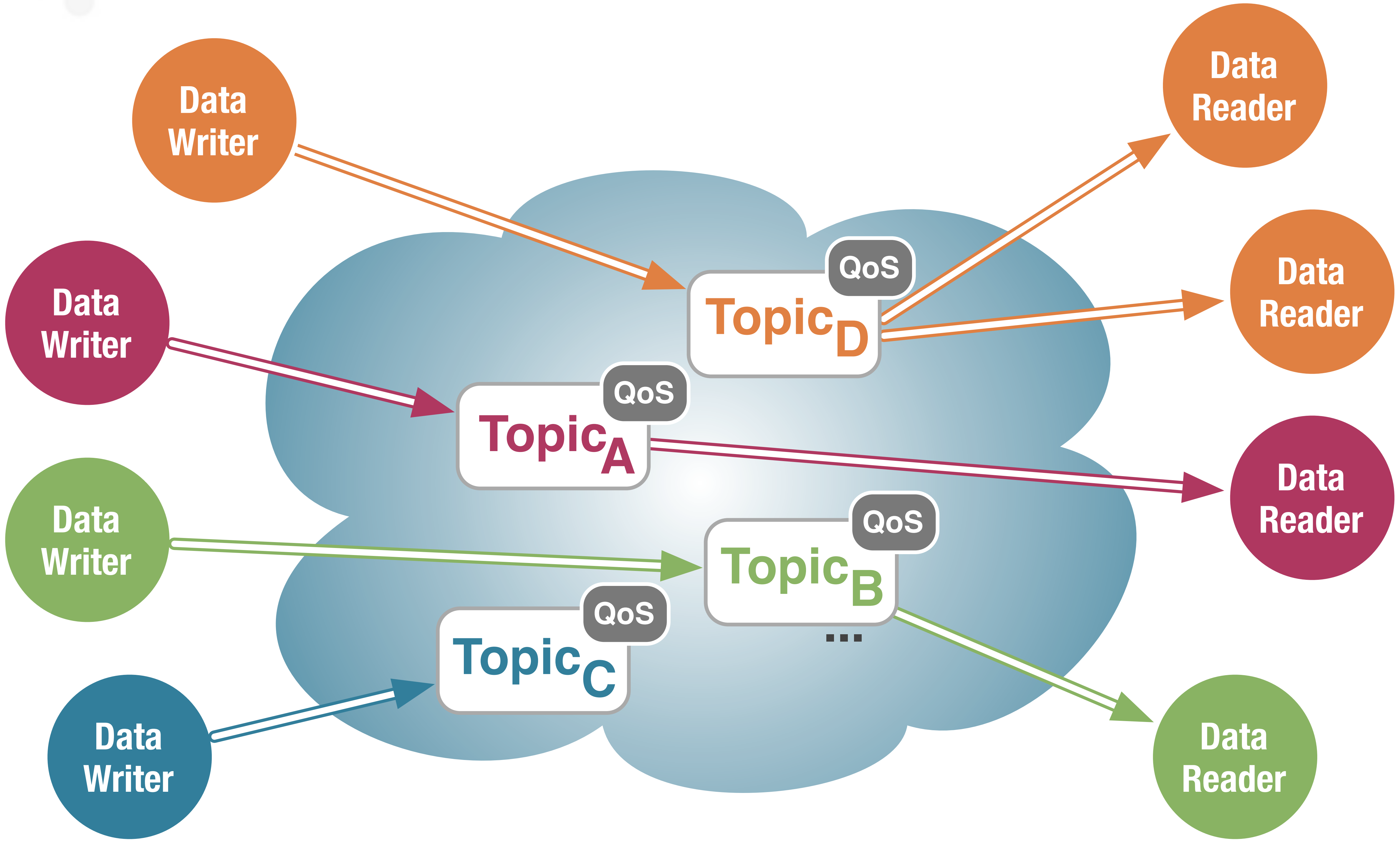


DDS Global Data Space

Copyright PrismTech, 2017

DDS Abstraction

Applications coordinate by **autonomously** and **asynchronously reading** and **writing** data in the Data Space enjoying **spatial** and **temporal decoupling**



DDS Global Data Space

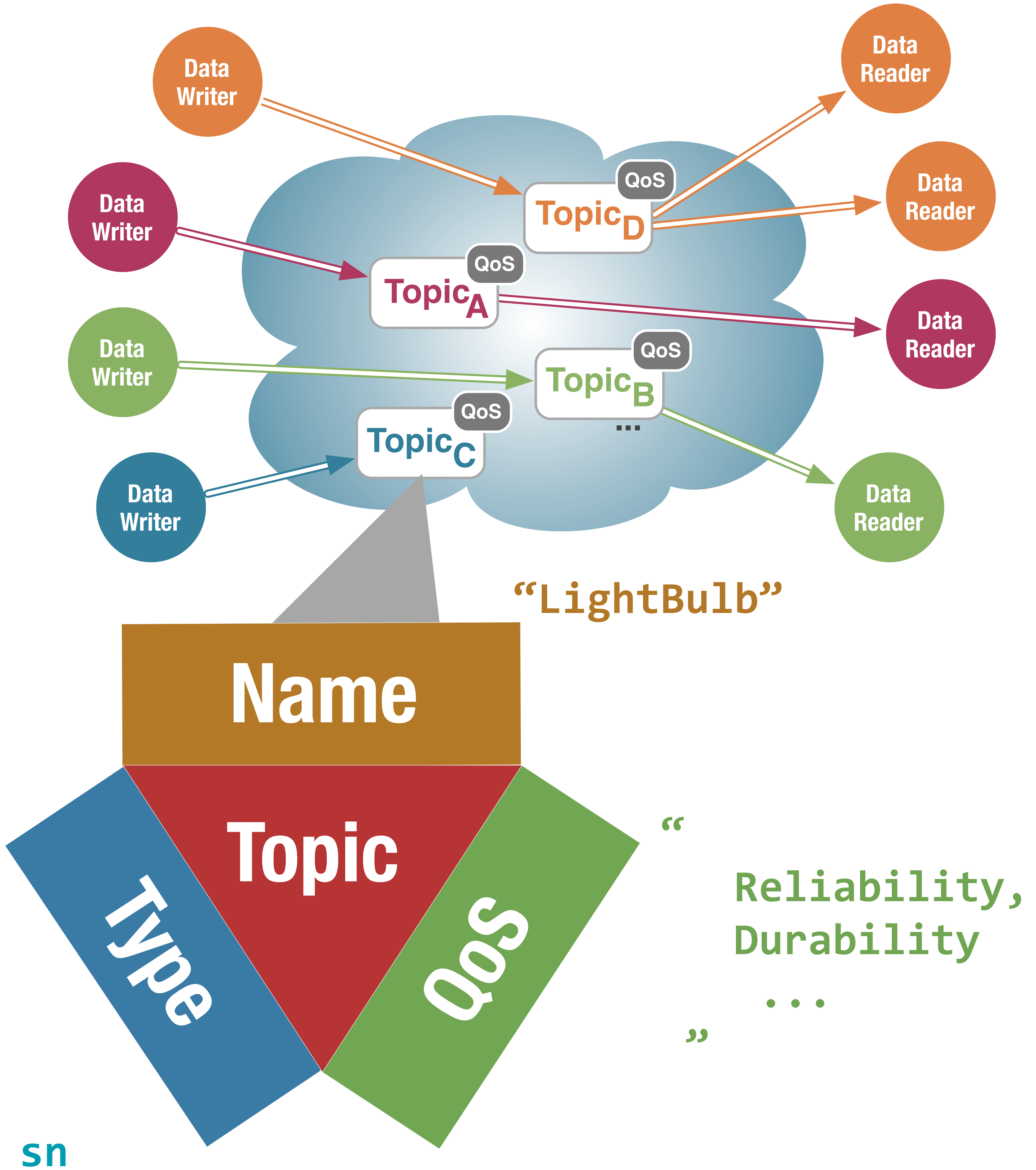
Copyright PrismTech, 2017

Topics

DDS data streams are defined by means of **Topics**

A **Topic** represented is by means of a <name, type, qos>

```
struct LightBulbState {
   string sn;
   float luminosity;
   long hue;
   boolean on;
};
#pragma keylist LightBulgState sn
```



"LightBulb"

" Reliability, Durability ... "

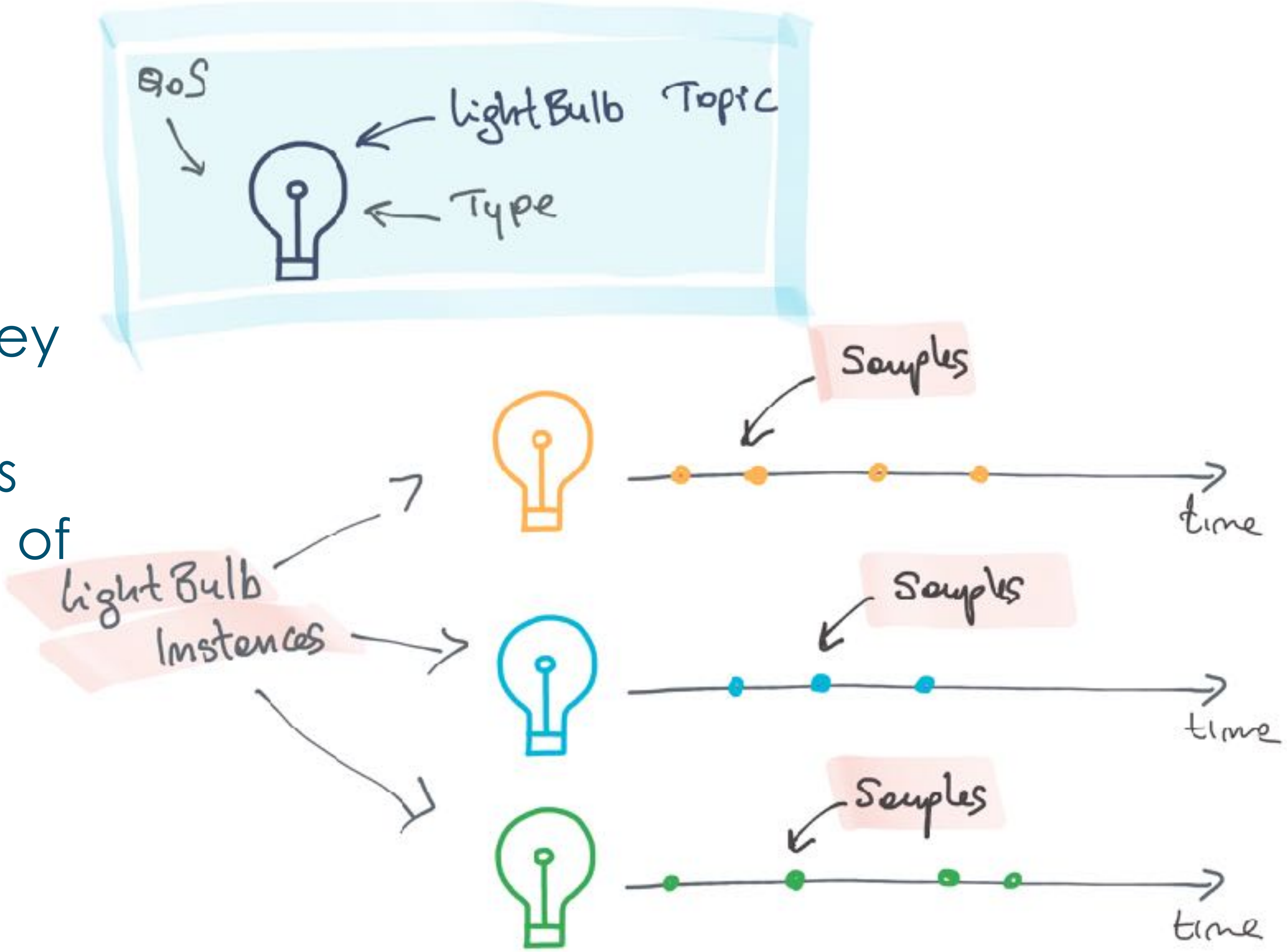
Copyright PrismTech, 2017

Topic Instances

Topic may **mark some** of their associated type **attributes** as **key-fields**

Each unique key value (tuple of key attributes) identifies a Topic Instance. Each Topic Instance has associated a FIFO ordered stream of samples

DDS provides useful **instance life-cycle management** and samples demultiplexing



Copyright PrismTech, 2017

Topics and Relations

A **topic** cans be seen as defining a **relation**

```
struct LightBulbState {
   string sn;
   float luminosity;
   long hue;
   boolean on;
};
#pragma keylist LightBulgState sn
```

<u>sn</u>	luminosity	hue	on
a123-21ef	0.5	12750	TRUE
600d-caf3	0.8	46920	FALSE
1234-c001	0.75	25500	TRUE

Copyright PrismTech, 2017

DDS Entities

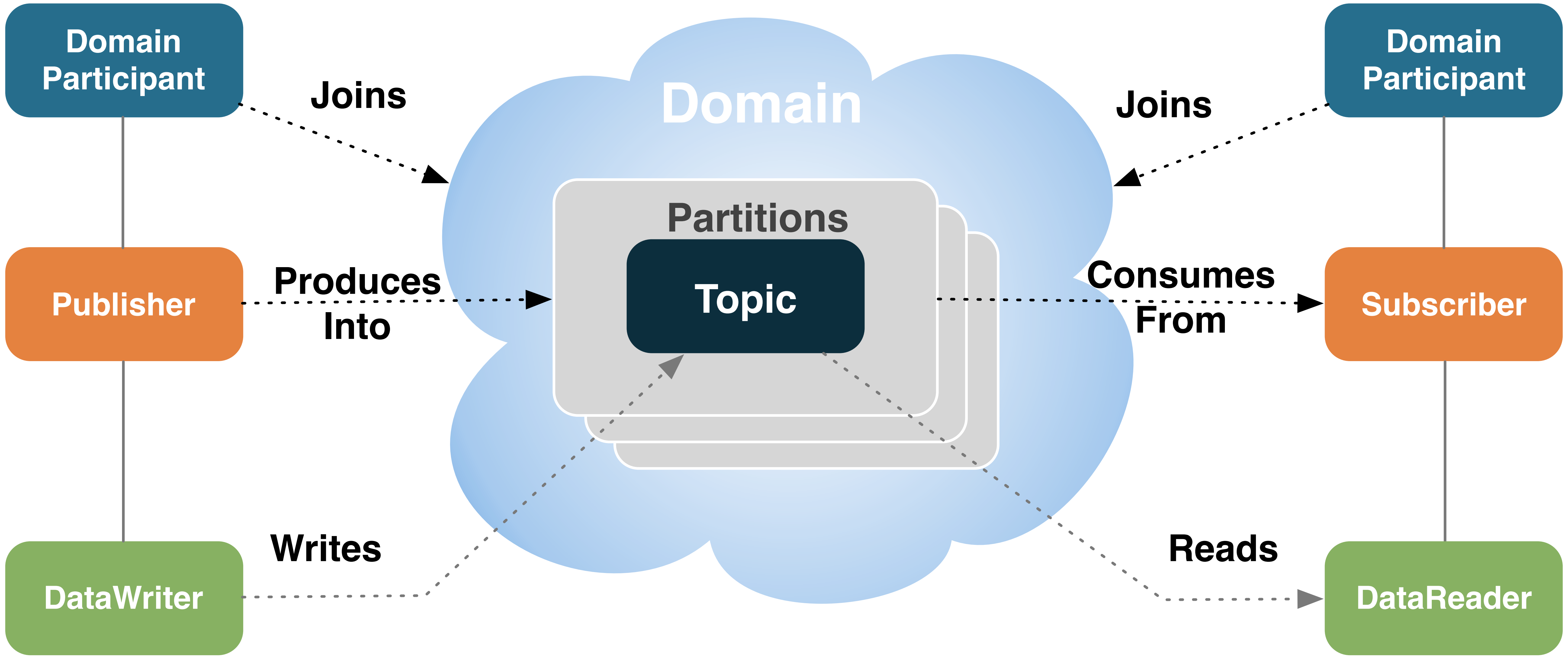
Copyright PrismTech, 2017

DDS Entities

DDS provides three different entities to control **where** and **what** data is read/written

The **DomainParticipant**, **Publisher** and **Subscriber** relate to the "**where**"

DataReader and **DataWriter** relate to the "**what**"

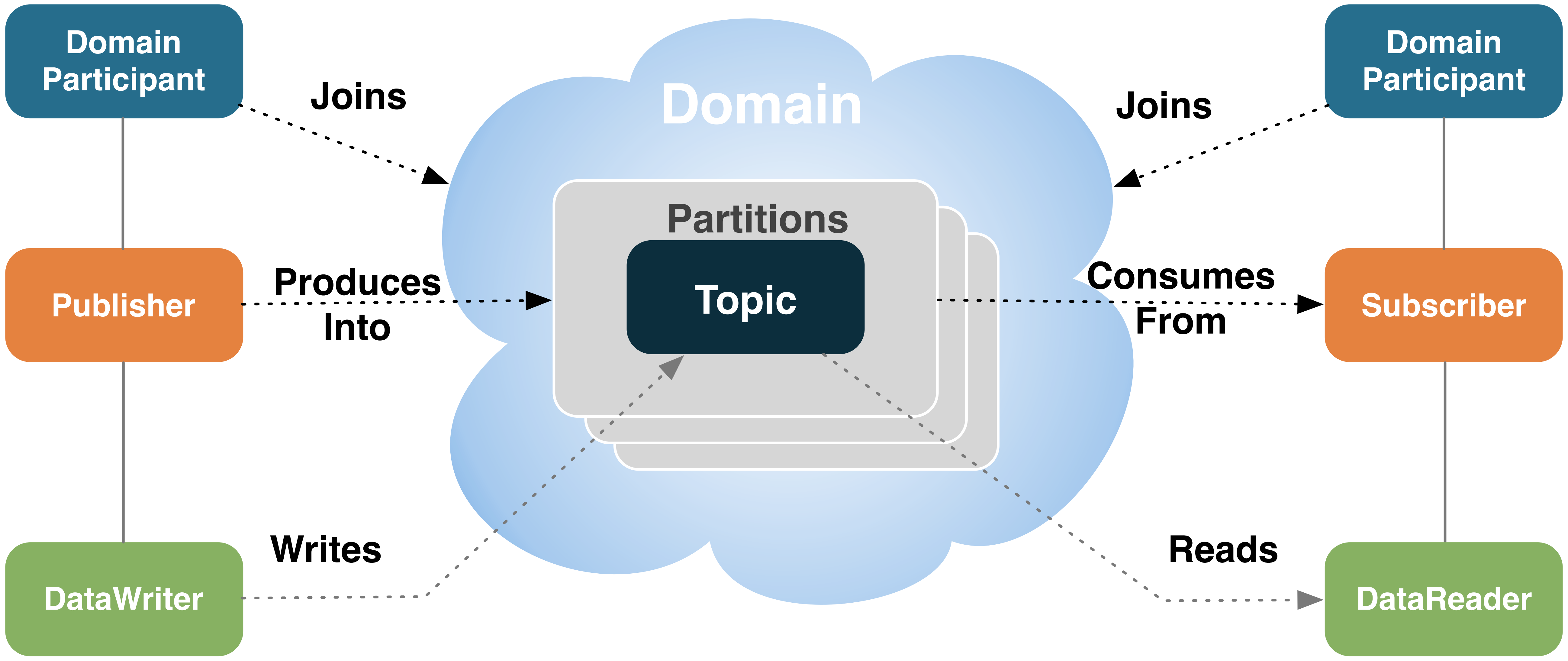


Copyright PrismTech, 2017

DDS Entities

DDS **QoS Policies** control, at a large extent, to the **"how" data** is **shared**

QoS Policies also control **resource utilisation**



Copyright PrismTech, 2017

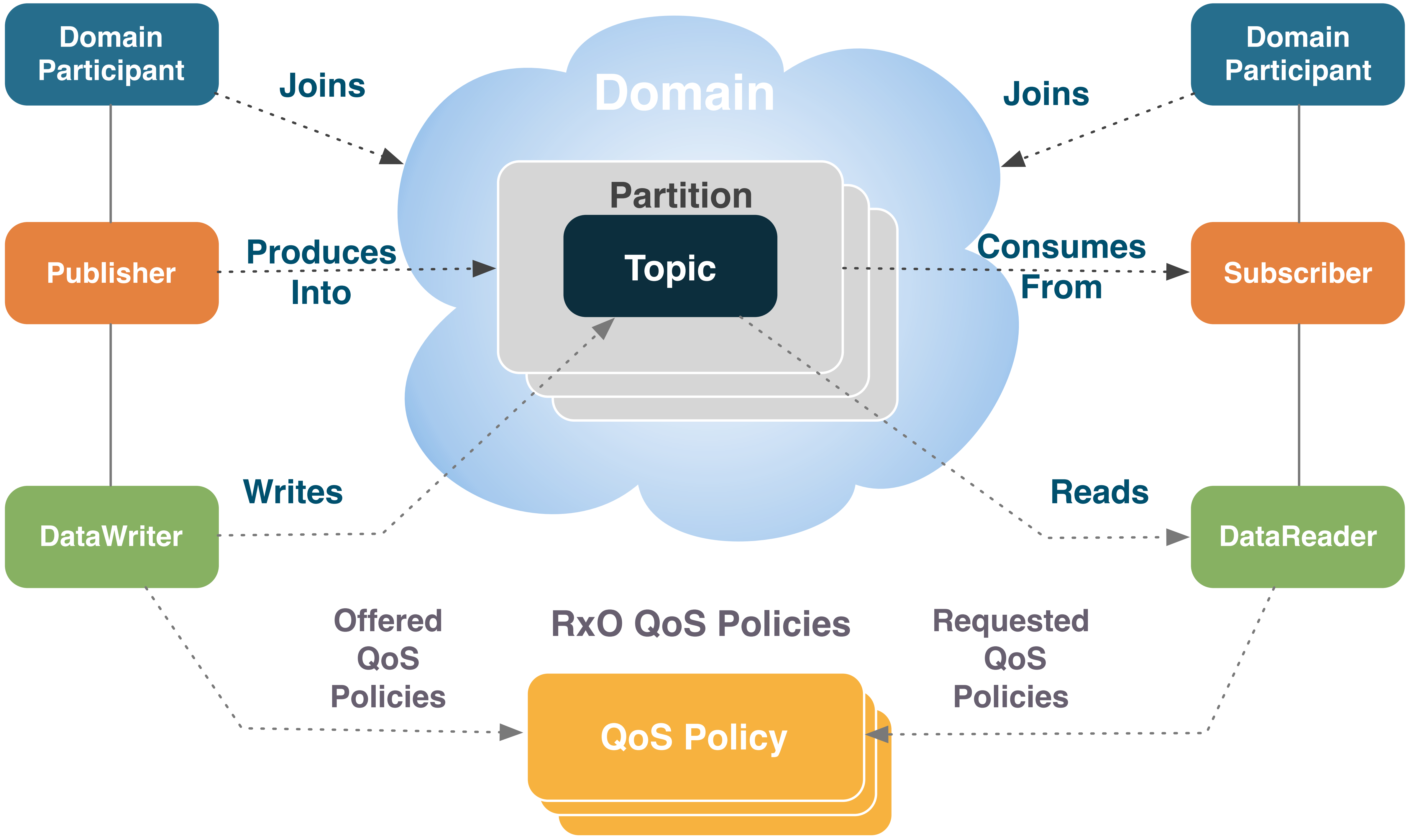
Matching Model

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

The **DR** and **DW** 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



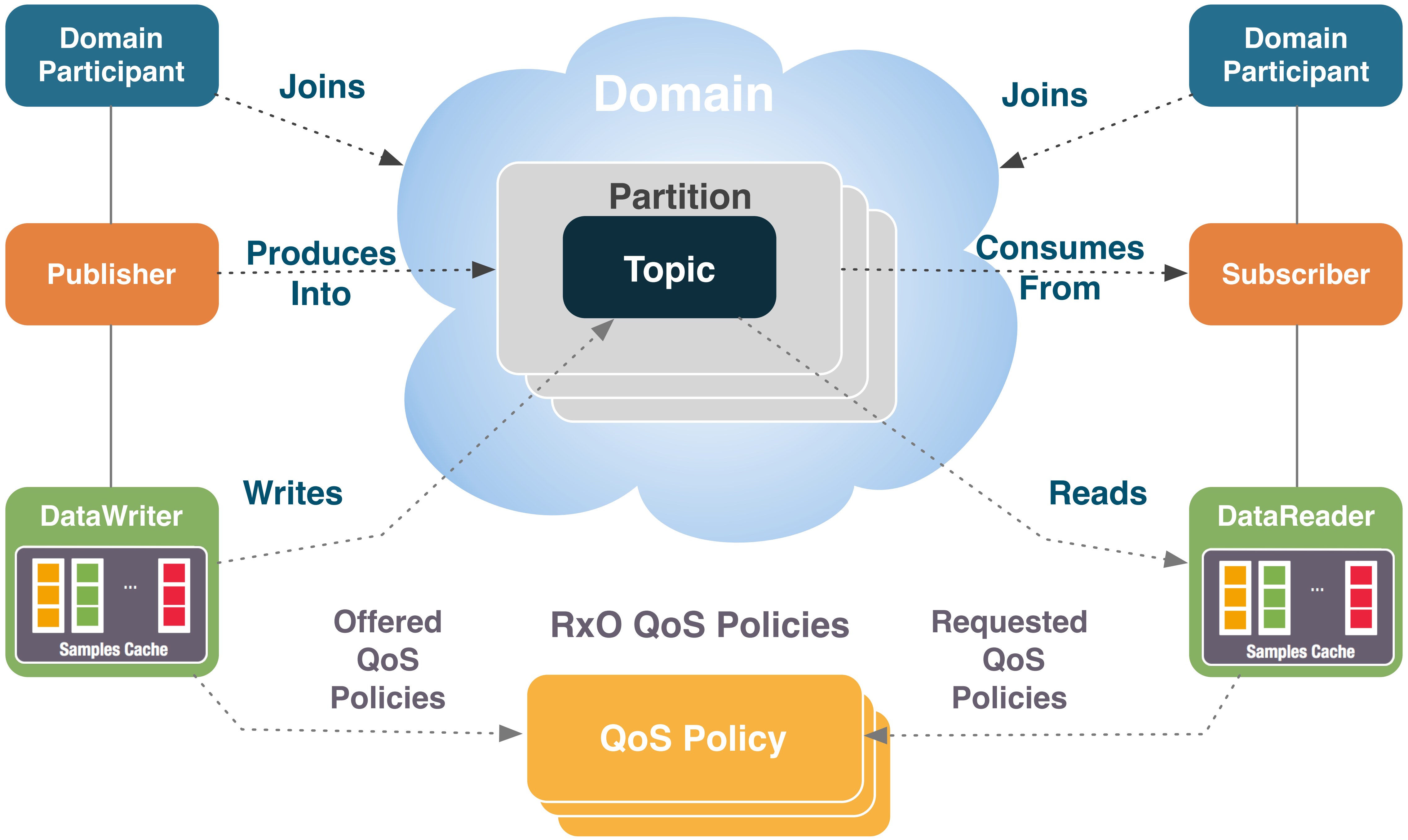
Copyright PrismTech, 2017

Storage Model

DataWriter and DataReaders have an associated samples cache

In a sense, what DDS does is to project, eventually, the relevant content of the writer cache into matching reader caches

As a consequence of these caches reads and writes are always local and non-blocking*



Copyright PrismTech, 2017

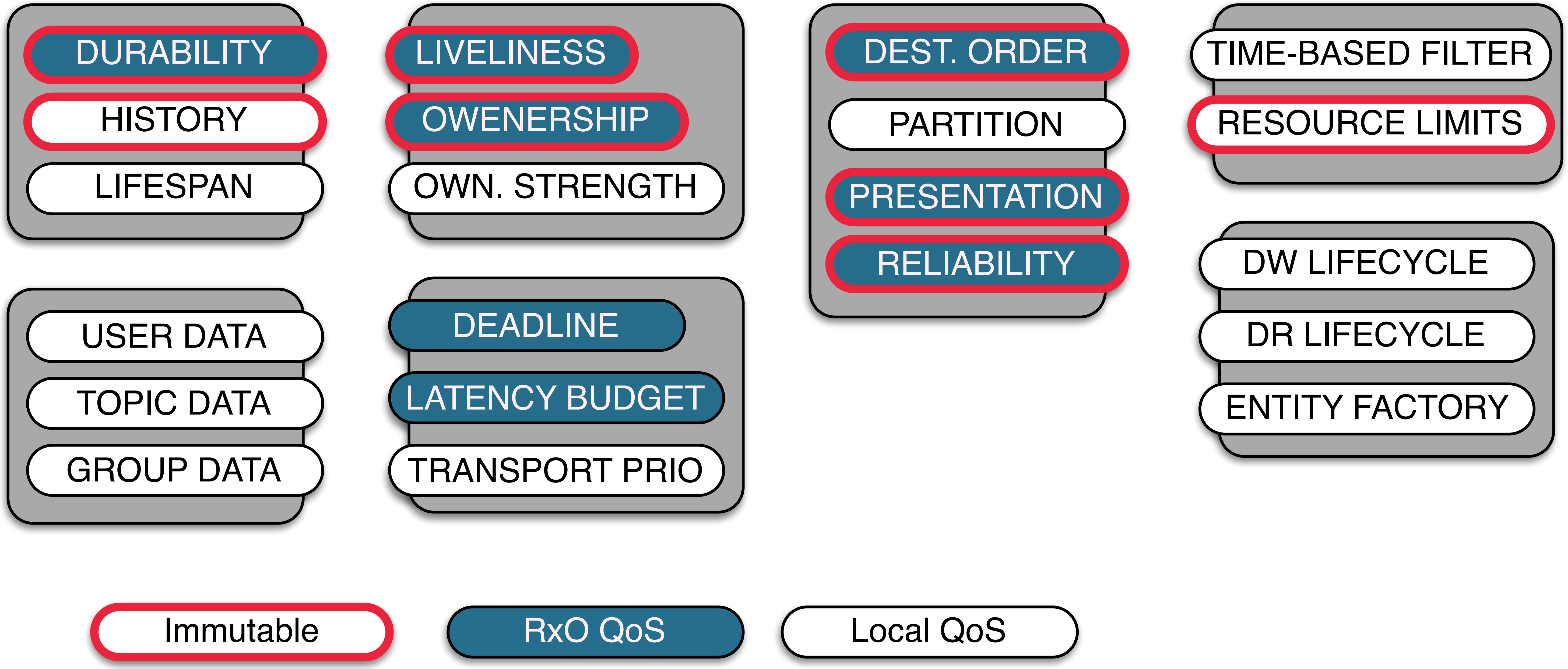
* **reads** never block and a write will only block, depending on QoS settings if sufficient resources are not available

QoS Policies

DDS QoS Policies

DDS provides 20+ standard QoS Policies

While this may seem a lot of complexity they are often used in combination to achieve certain patterns



DURABILITY
HISTORY
LIFESPAN

USER DATA
TOPIC DATA
GROUP DATA

LIVELINESS
OWENERSHIP
OWN. STRENGTH

DEADLINE
LATENCY BUDGET
TRANSPORT PRIO

DEST. ORDER
PARTITION
PRESENTATION
RELIABILITY

TIME-BASED FILTER
RESOURCE LIMITS

DW LIFECYCLE
DR LIFECYCLE
ENTITY FACTORY

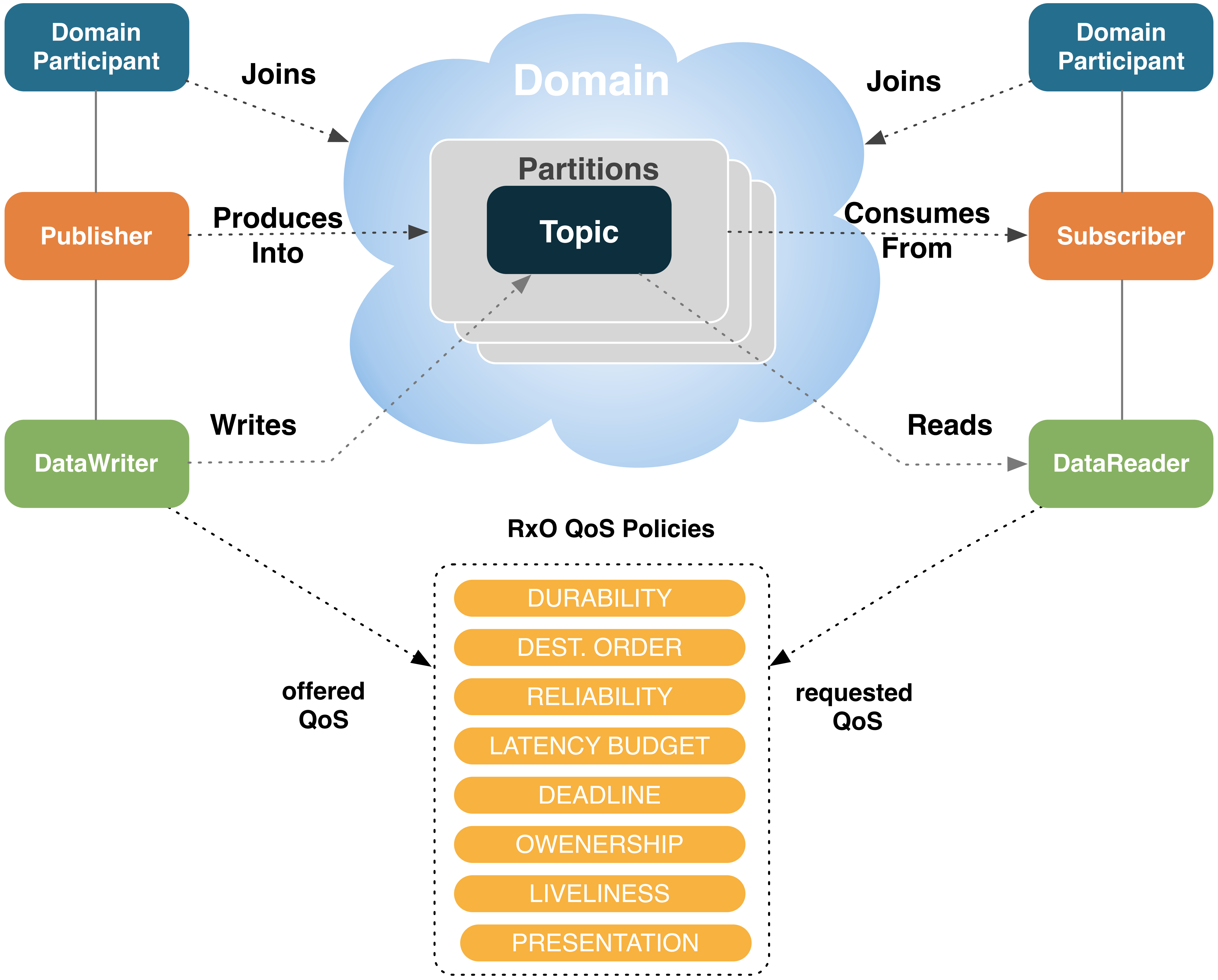
Immutable
RxO QoS
Local QoS

Copyright PrismTech, 2017

RxO Model

A subset of the QoS Policies impact the matching

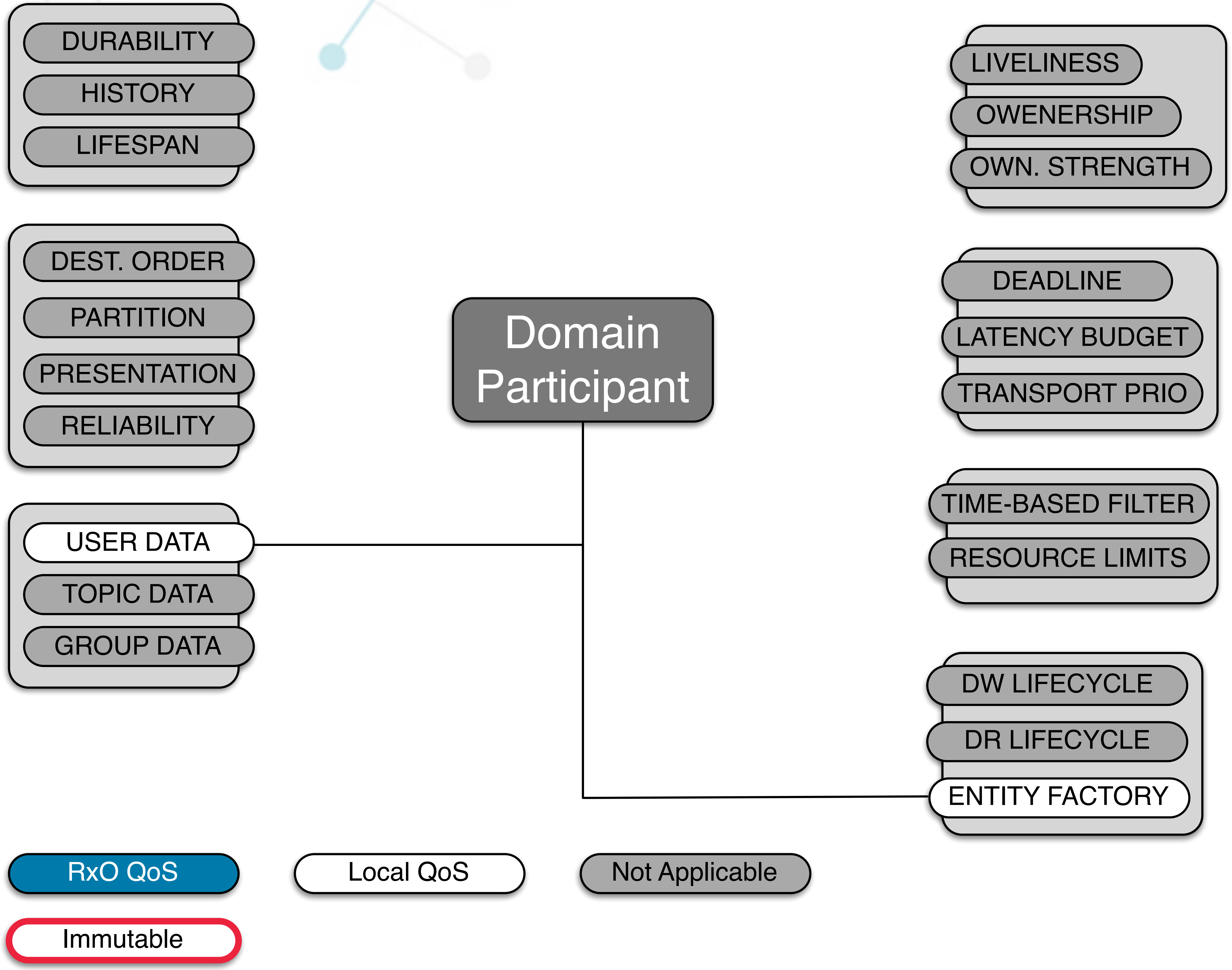
Most of thee QoS relate to end-to-end properties of data



Copyright PrismTech, 2017

QoS Policies

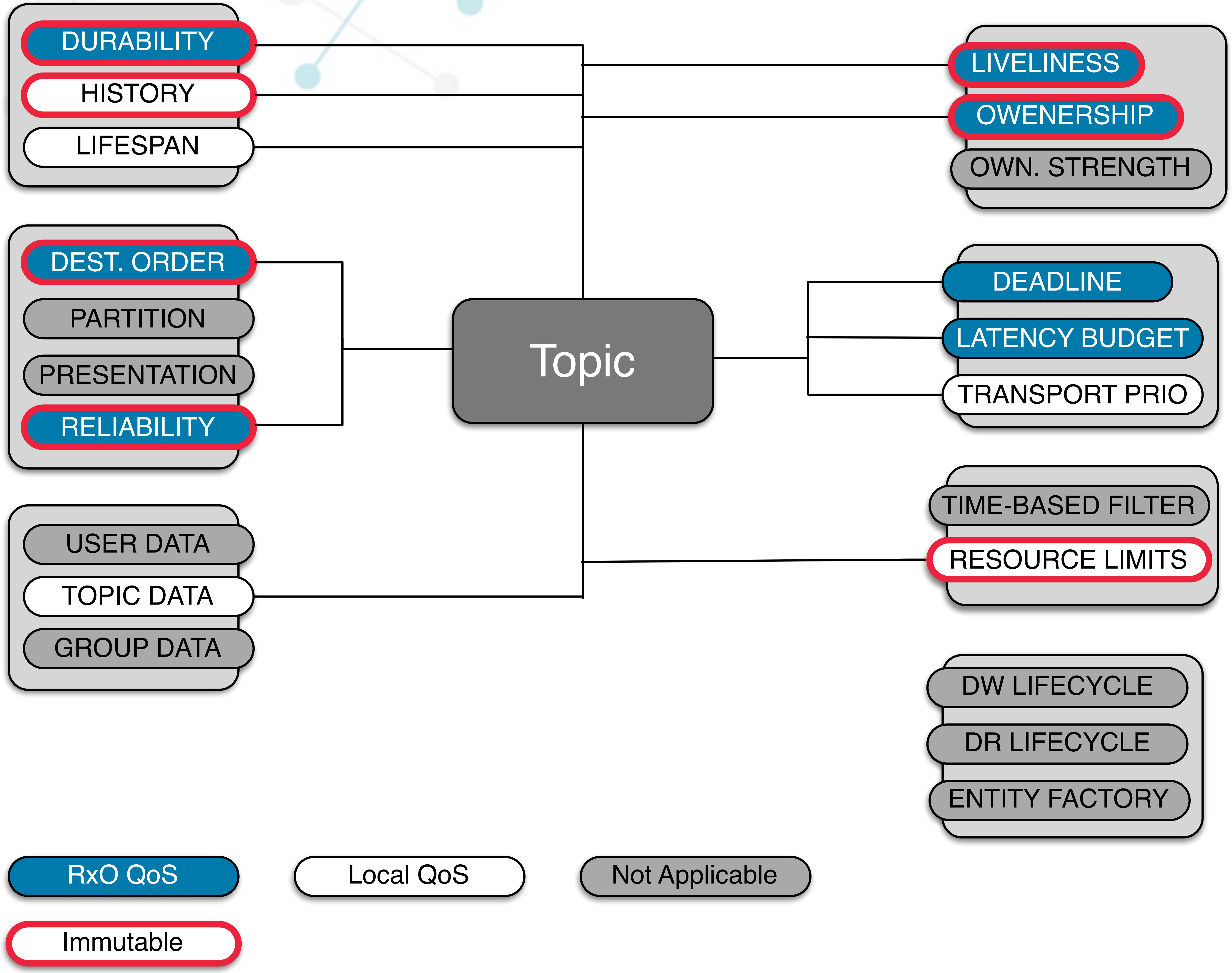
Domain Participant



Copyright PrismTech, 2017

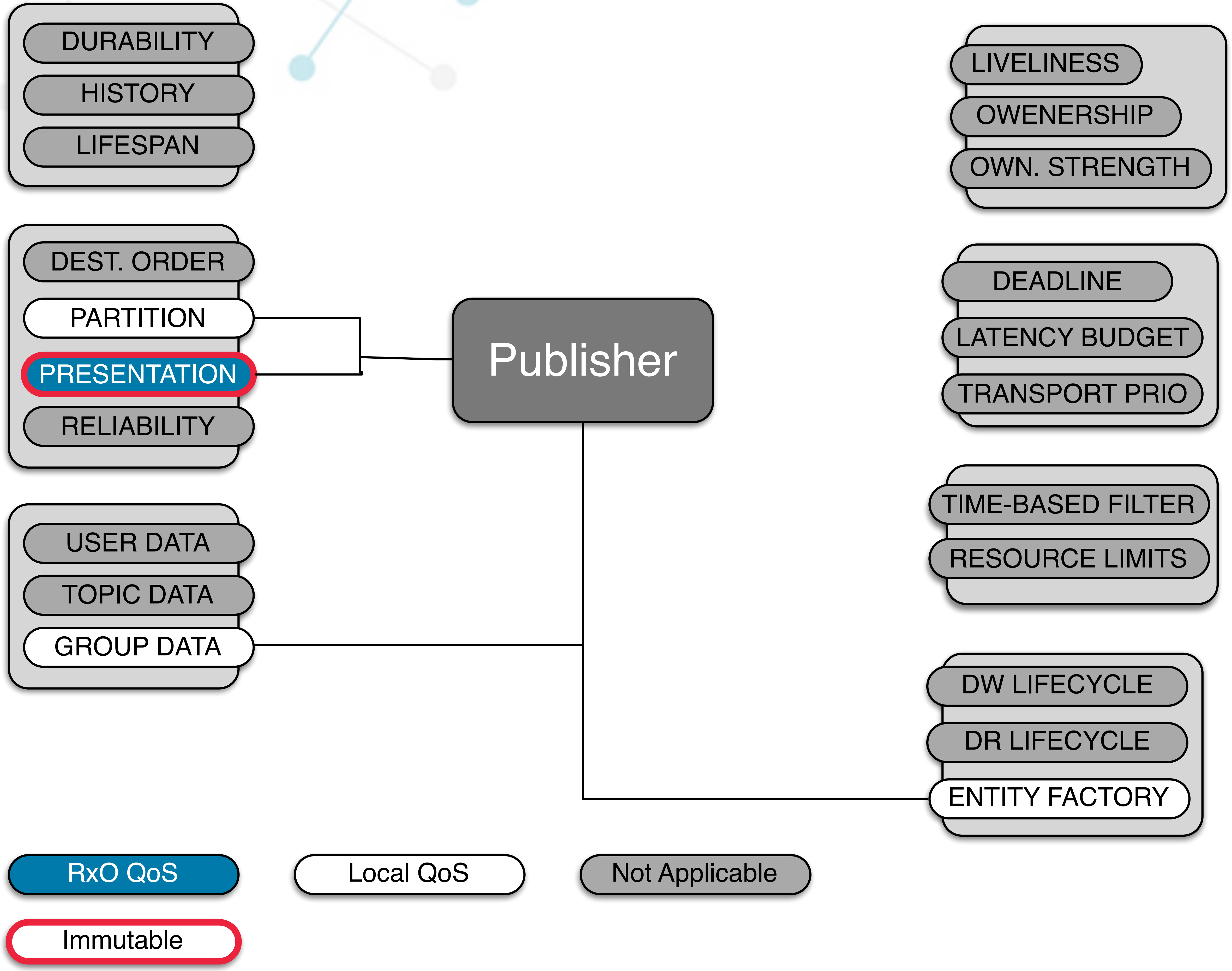
QoS Policies

Topic



Copyright PrismTech, 2017

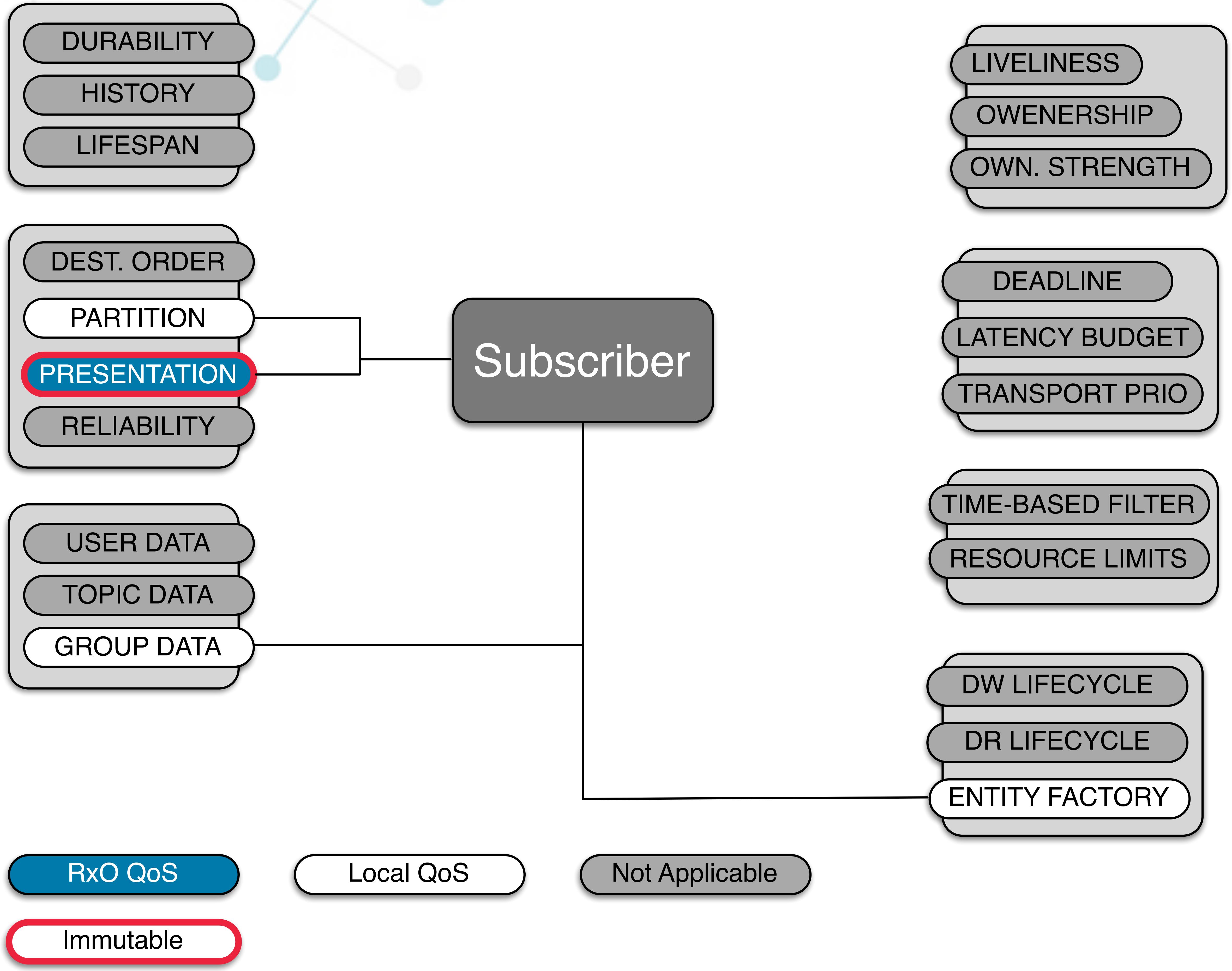
QoS Policies
Publisher



Copyright PrismTech, 2017

QoS Policies

Subscriber



DURABILITY

HISTORY

LIFESPAN

DEST. ORDER

PARTITION

PRESENTATION

RELIABILITY

USER DATA

TOPIC DATA

GROUP DATA

Subscriber

LIVELINESS

OWENERSHIP

OWN. STRENGTH

DEADLINE

LATENCY BUDGET

TRANSPORT PRIO

TIME-BASED FILTER

RESOURCE LIMITS

DW LIFECYCLE

DR LIFECYCLE

ENTITY FACTORY

RxO QoS

Local QoS

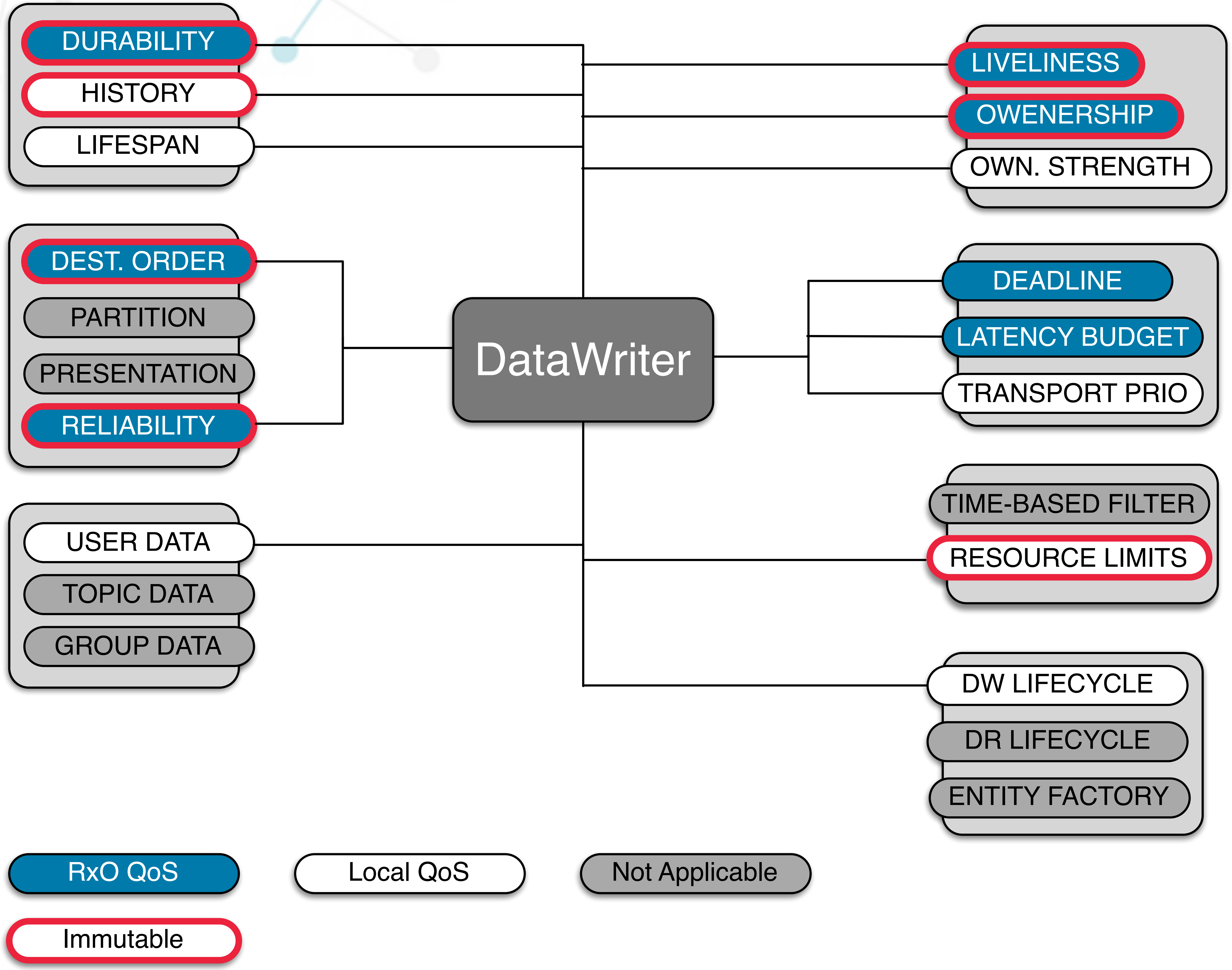
Not Applicable

Immutable

Copyright PrismTech, 2017

QoS Policies

DataWriter



DURABILITY
HISTORY
LIFESPAN

DEST. ORDER
PARTITION
PRESENTATION
RELIABILITY

USER DATA
TOPIC DATA
GROUP DATA

DataWriter

LIVELINESS
OWENERSHIP
OWN. STRENGTH

DEADLINE
LATENCY BUDGET
TRANSPORT PRIO

TIME-BASED FILTER
RESOURCE LIMITS

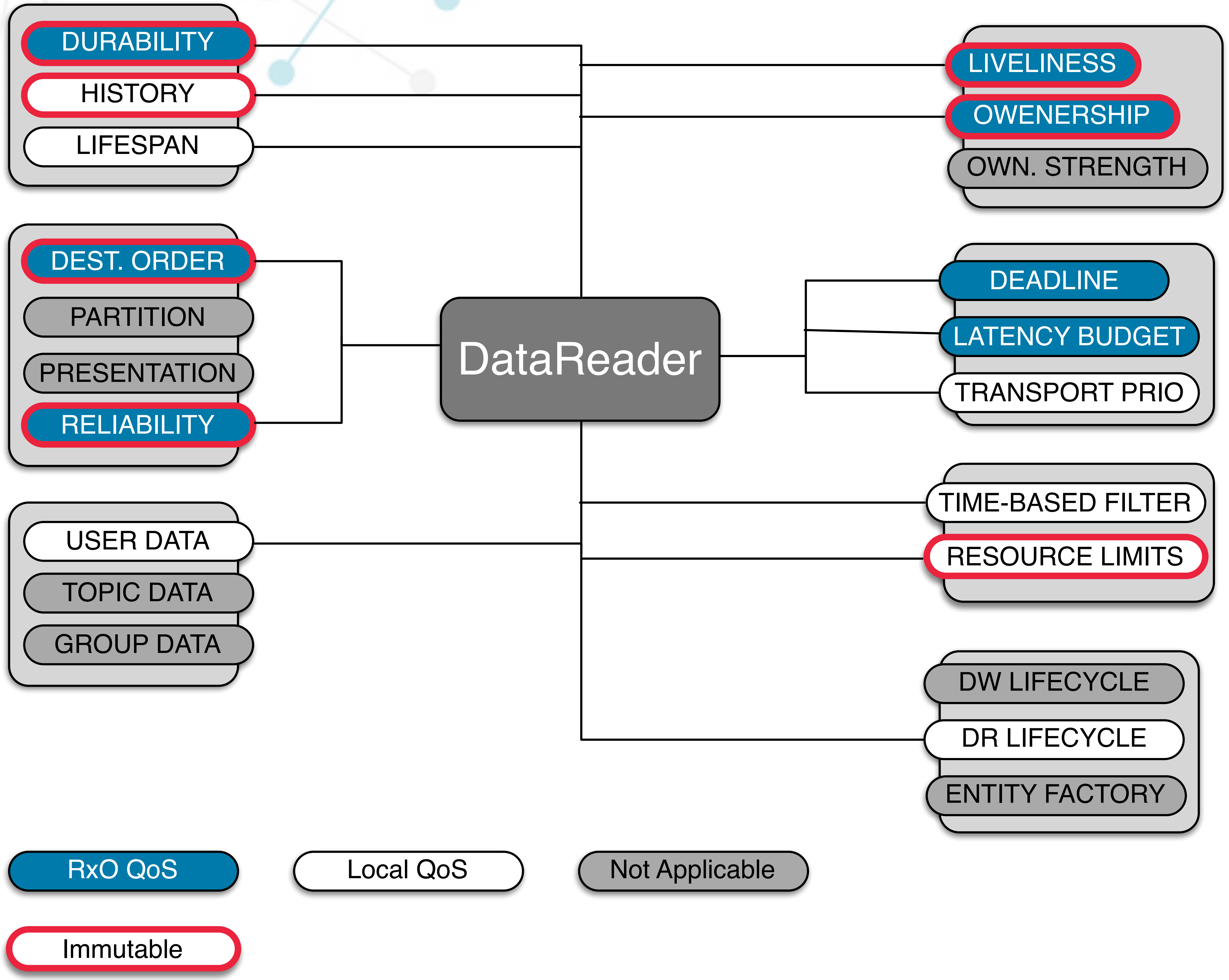
DW LIFECYCLE
DR LIFECYCLE
ENTITY FACTORY

RxO QoS
Local QoS
Not Applicable
Immutable

Copyright PrismTech, 2017

QoS Policies

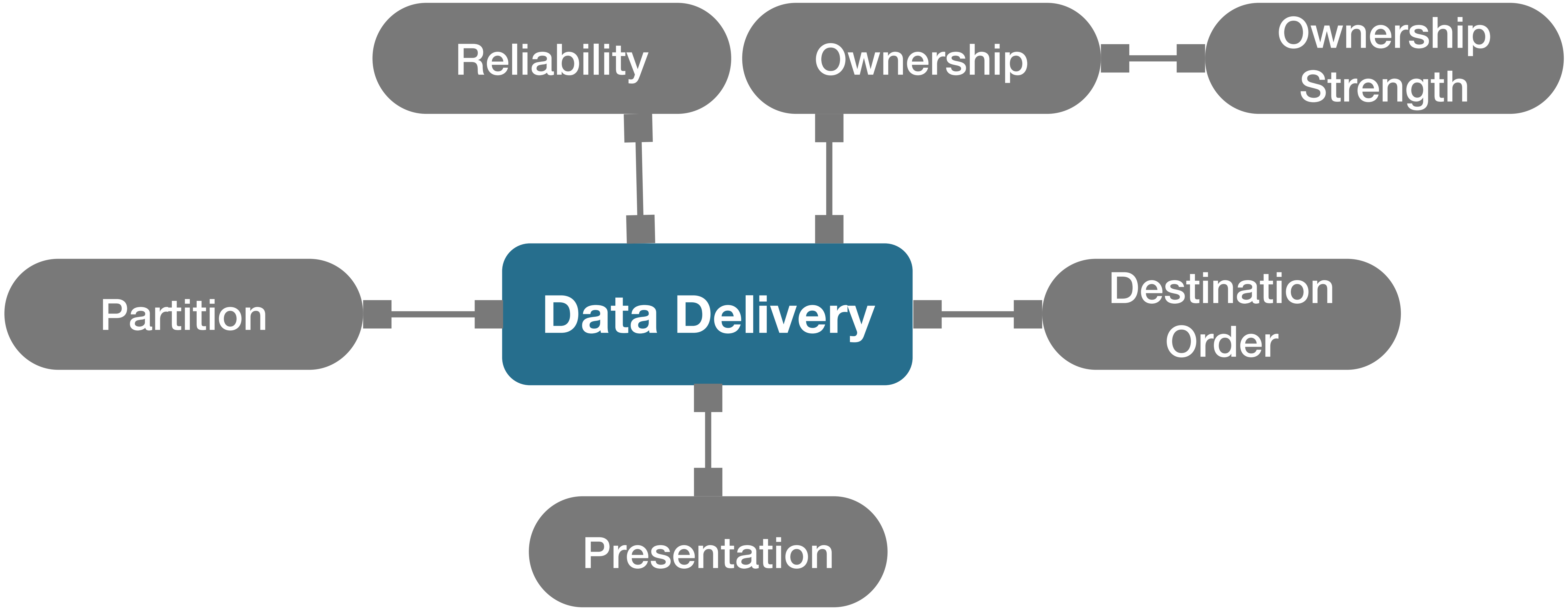
Data Reader



Copyright PrismTech, 2017

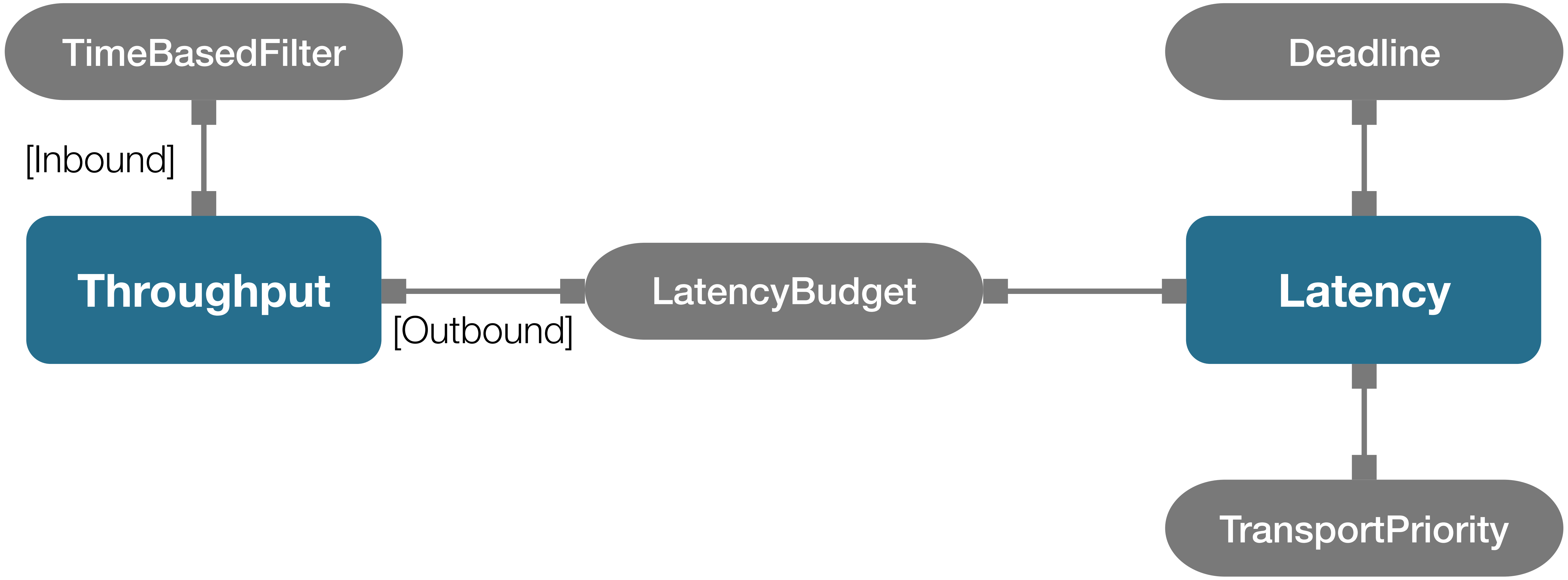
QoS Categorisation

Data Delivery



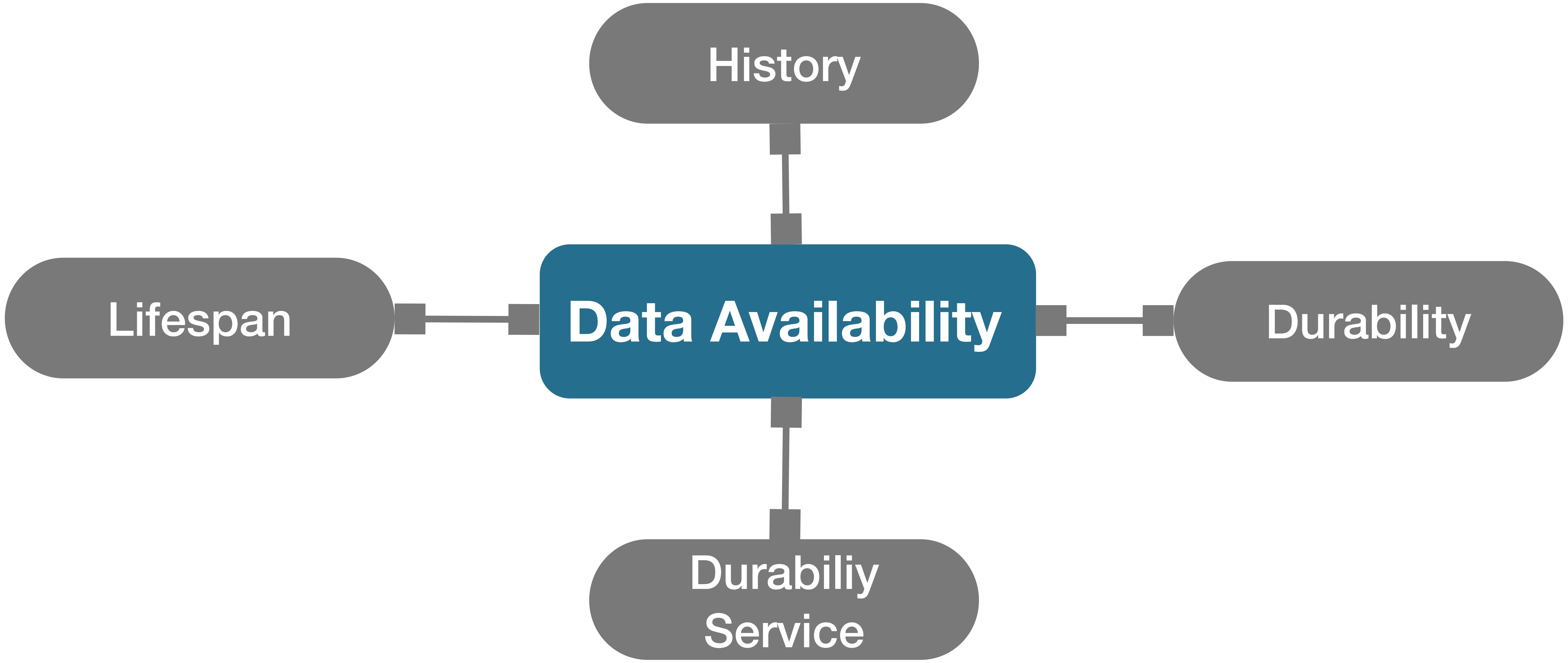
Copyright PrismTech, 2017

Temporal Properties



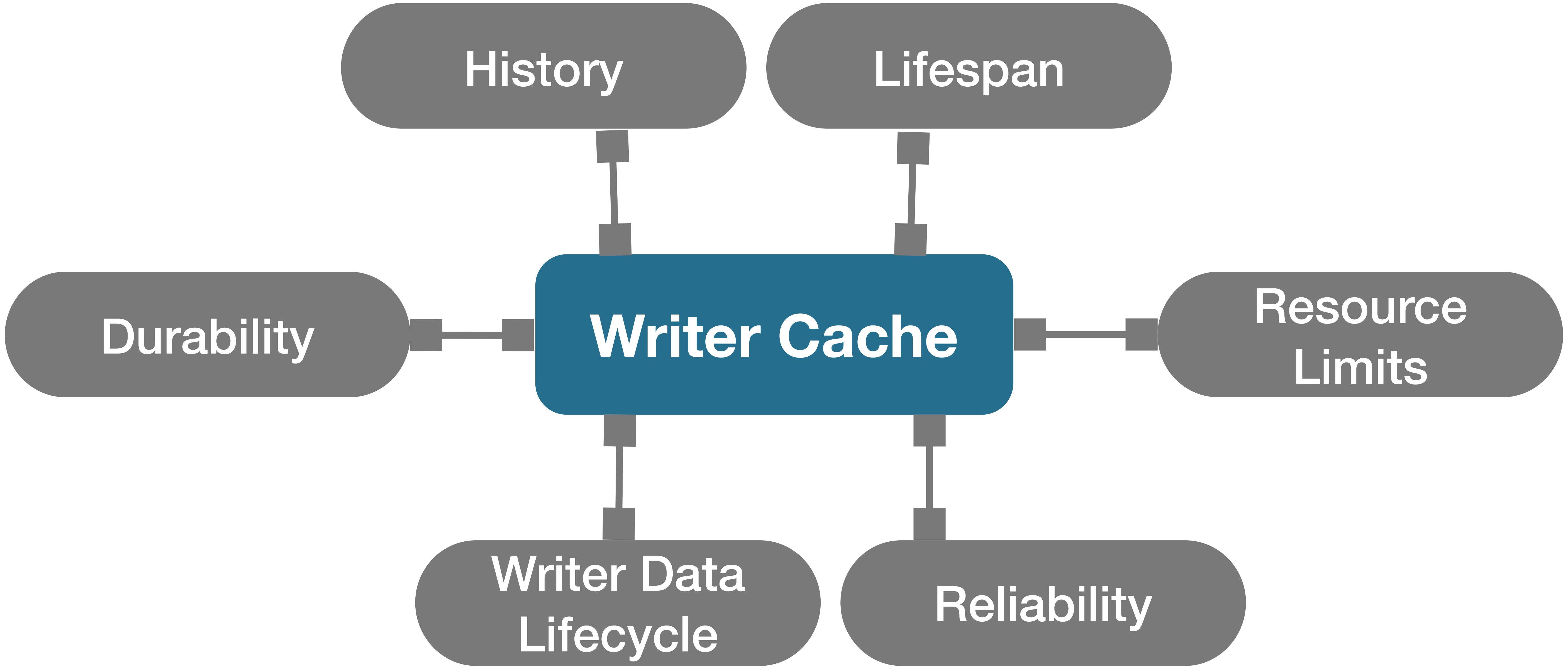
Copyright PrismTech, 2017

Data Availability



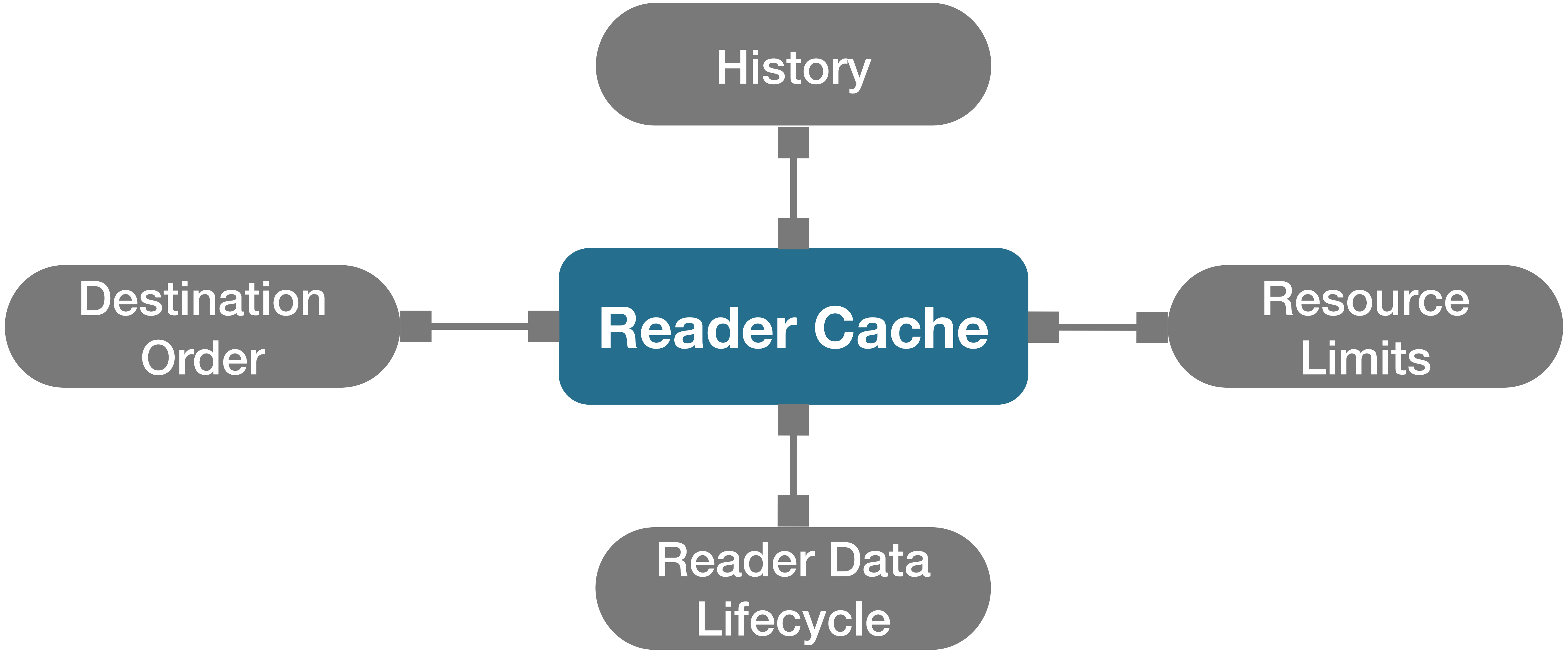
Copyright PrismTech, 2017

Writer Cache



Copyright PrismTech, 2017

Reader Cache



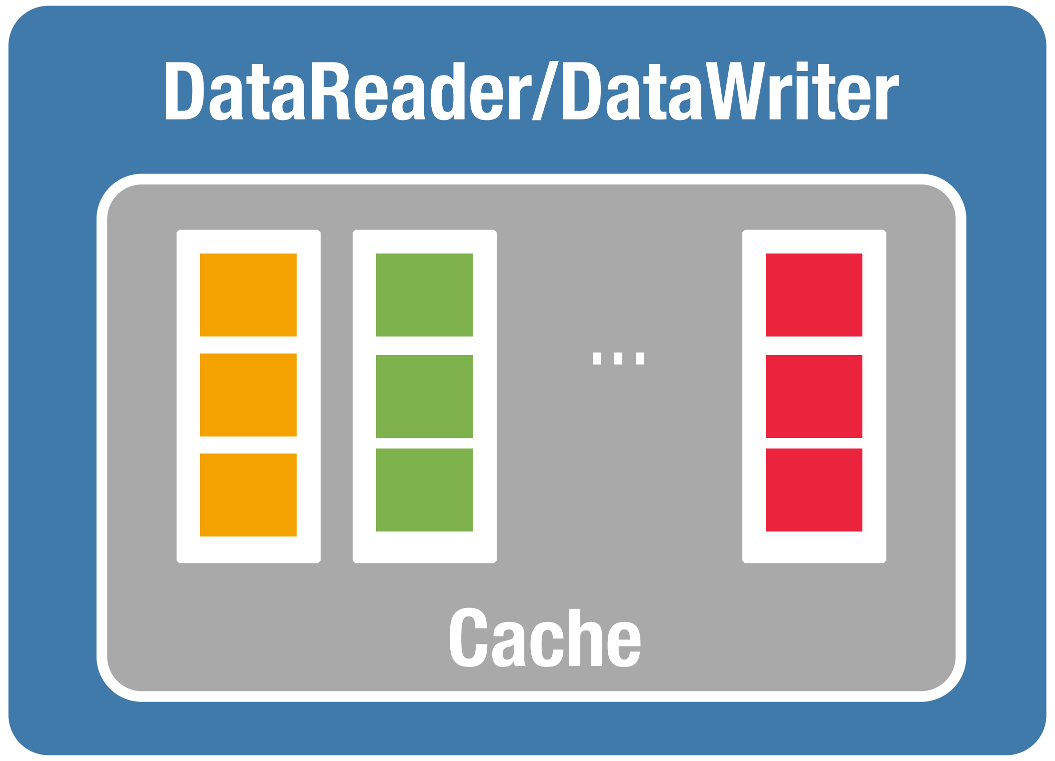
Copyright PrismTech, 2017

QoS Semantics

History

History

The HISTORY QoS Policy provide controls on the replacement strategy and depth of the cache

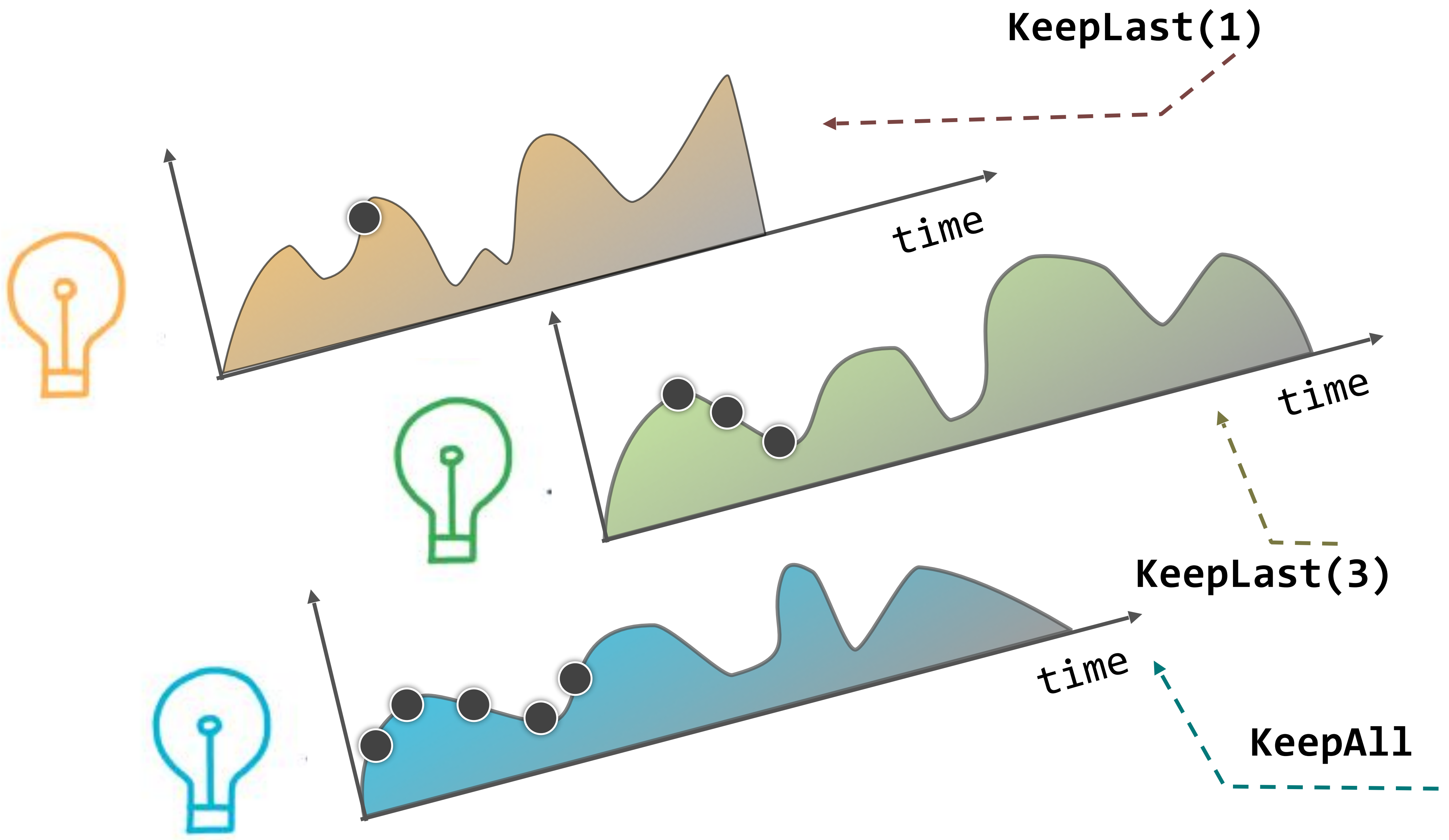


QoS Policy	Applicability	RxO	Modifiable
HISTORY	T, DR, DW	N	N

Copyright PrismTech, 2017

Data Writer History

The **DataWriter** HISTORY QoS Policy controls the amount of data that can be made available to late joining DataReaders under `TRANSIENT_LOCAL` Durability



QoS Policy	Applicability	RxO	Modifiable
HISTORY	T, DR, DW	N	N

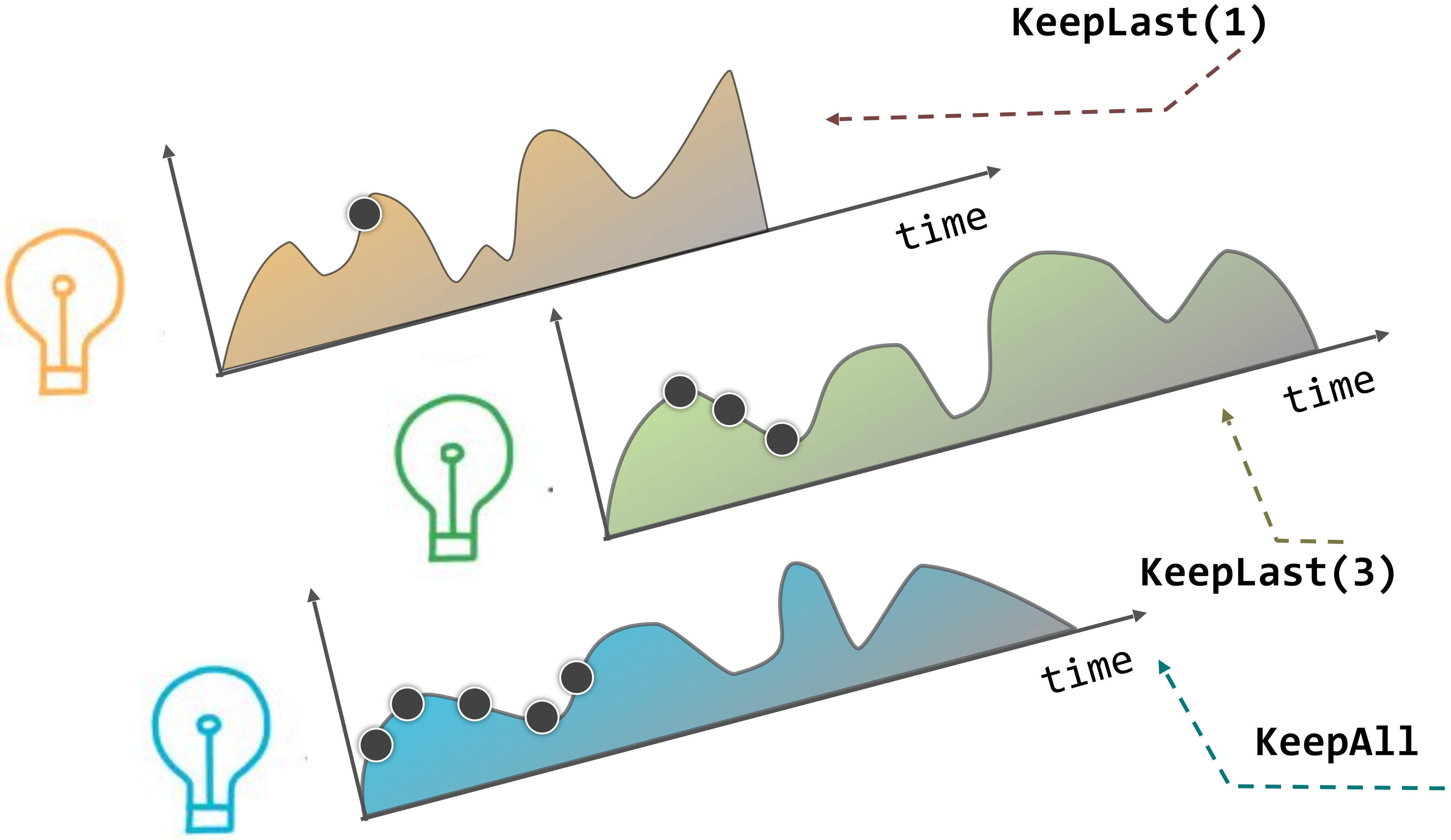
Copyright PrismTech, 2017

DataReader History

The **DataReader** HISTORY QoS Policy controls how many samples will be kept on the reader cache

Keep Last. DDS will keep the most recent “depth” samples of each instance of data identified by its key

Keep All. The DDS keep all the samples of each instance of data identified by its key -- up to reaching some configurable resource limits



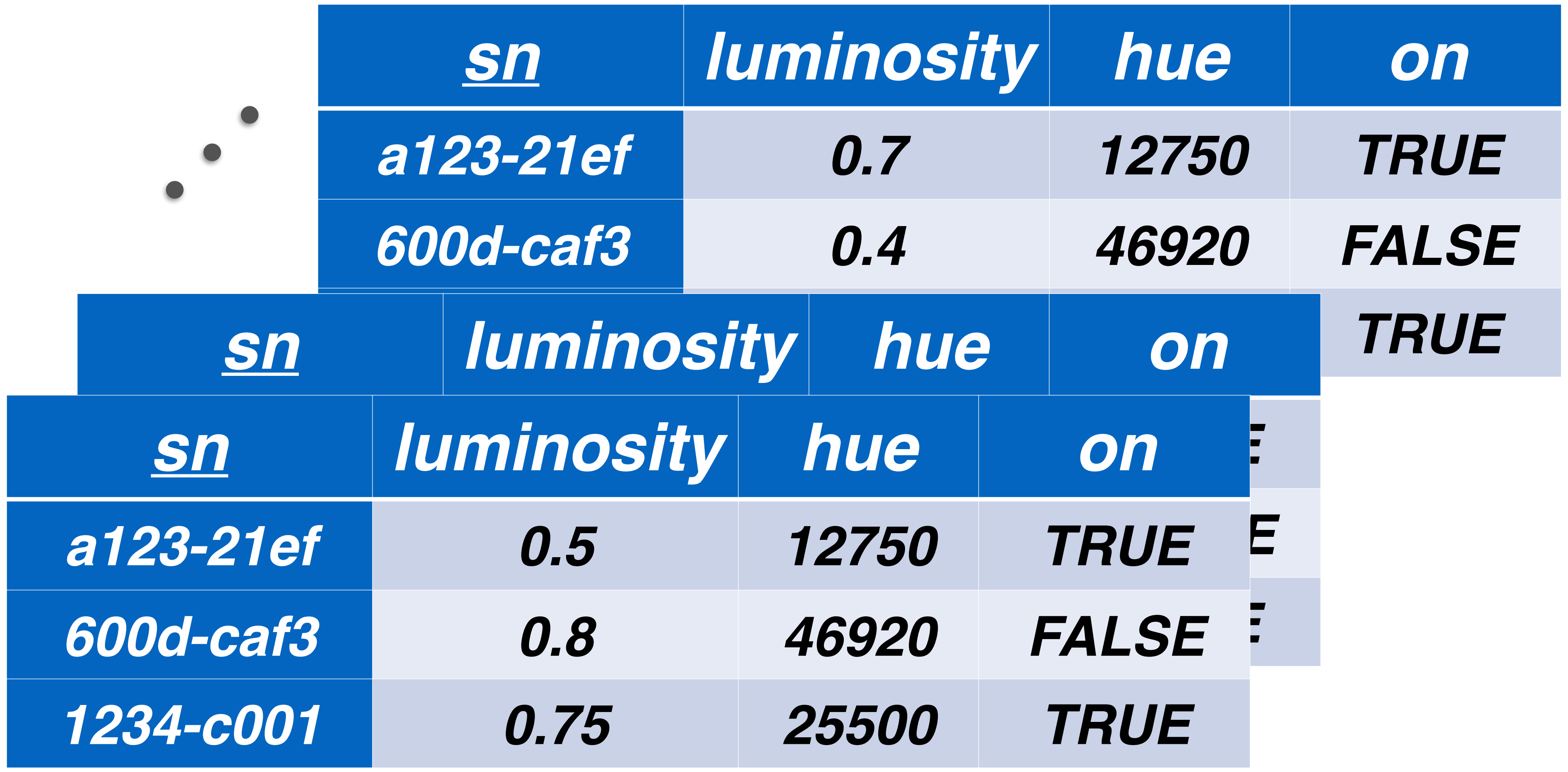
QoS Policy	Applicability	RxO	Modifiable
HISTORY	T, DR, DW	N	N

Copyright PrismTech, 2017

History: Another Perspective

As we saw earlier a topic can be seen as a relation (in the sense of relational algebra)

The history makes the table a cube, where each slice of the cube represents a value of the relation at some point in time



sn	luminosity	hue	on
a123-21ef	0.7	12750	TRUE
600d-caf3	0.4	46920	FALSE
			TRUE

sn	luminosity	hue	on
			E
			E
			E

sn	luminosity	hue	on
a123-21ef	0.5	12750	TRUE
600d-caf3	0.8	46920	FALSE
1234-c001	0.75	25500	TRUE

Copyright PrismTech, 2017

Durability

Copyright PrismTech, 2017

Durability

The DURABILITY QoS controls the data availability w.r.t. late joiners, specifically the DDS provides the following variants:

Volatile. No need to keep data instances for late joining data readers

Transient Local. Data instance availability for late joining data reader is tied to the data writer availability

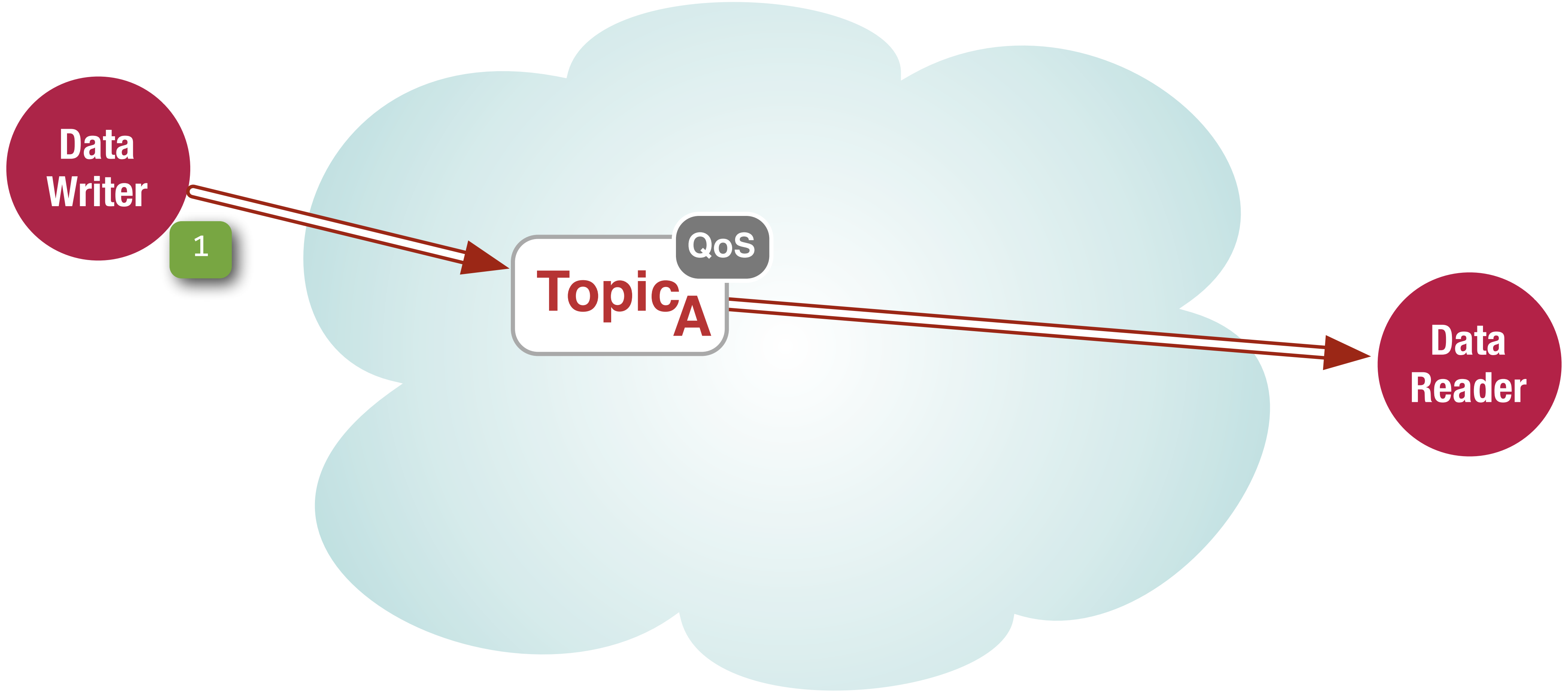
Transient. Data instance availability outlives the data writer

Persistent. Data instance availability outlives system restarts

QoS Policy	Applicability	RxO	Modifiable
DURABILITY	T, DR, DW	Y	N

Copyright PrismTech, 2017

Volatile Durability

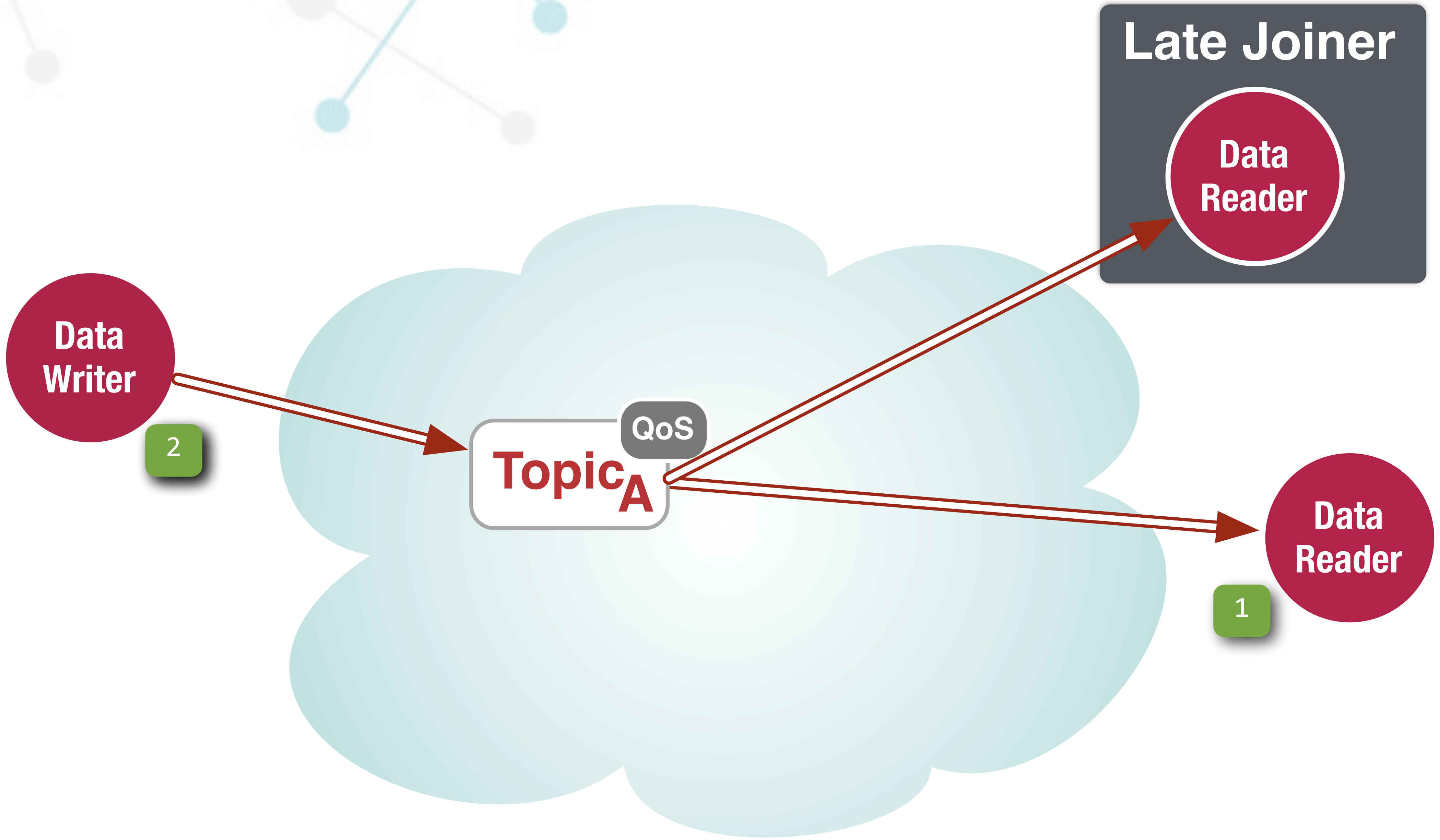


No Time Decoupling

Readers get only data produced after they joined the Global Data Space

Copyright PrismTech, 2017

Volatile Durability

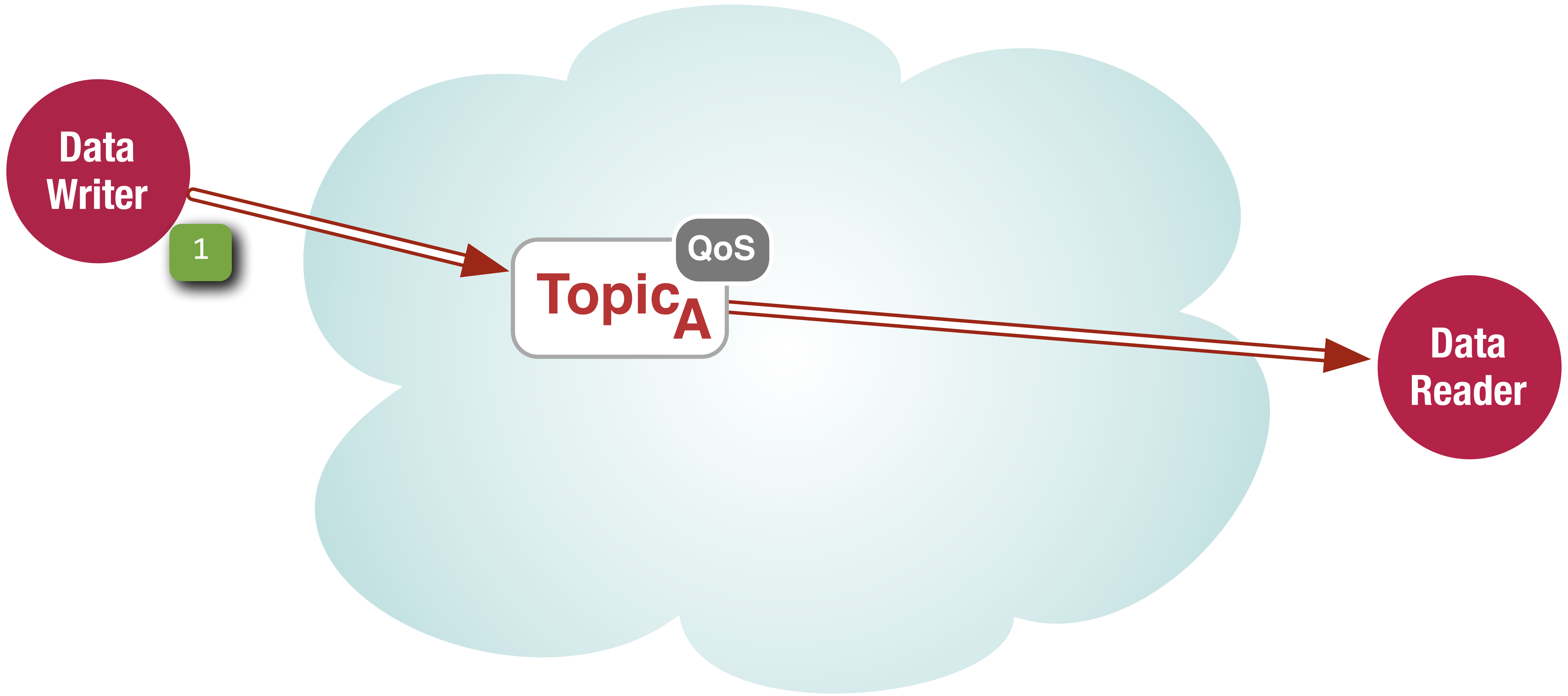


No Time Decoupling

Readers get only data produced after they joined the Global Data Space

Copyright PrismTech, 2017

Transient Local

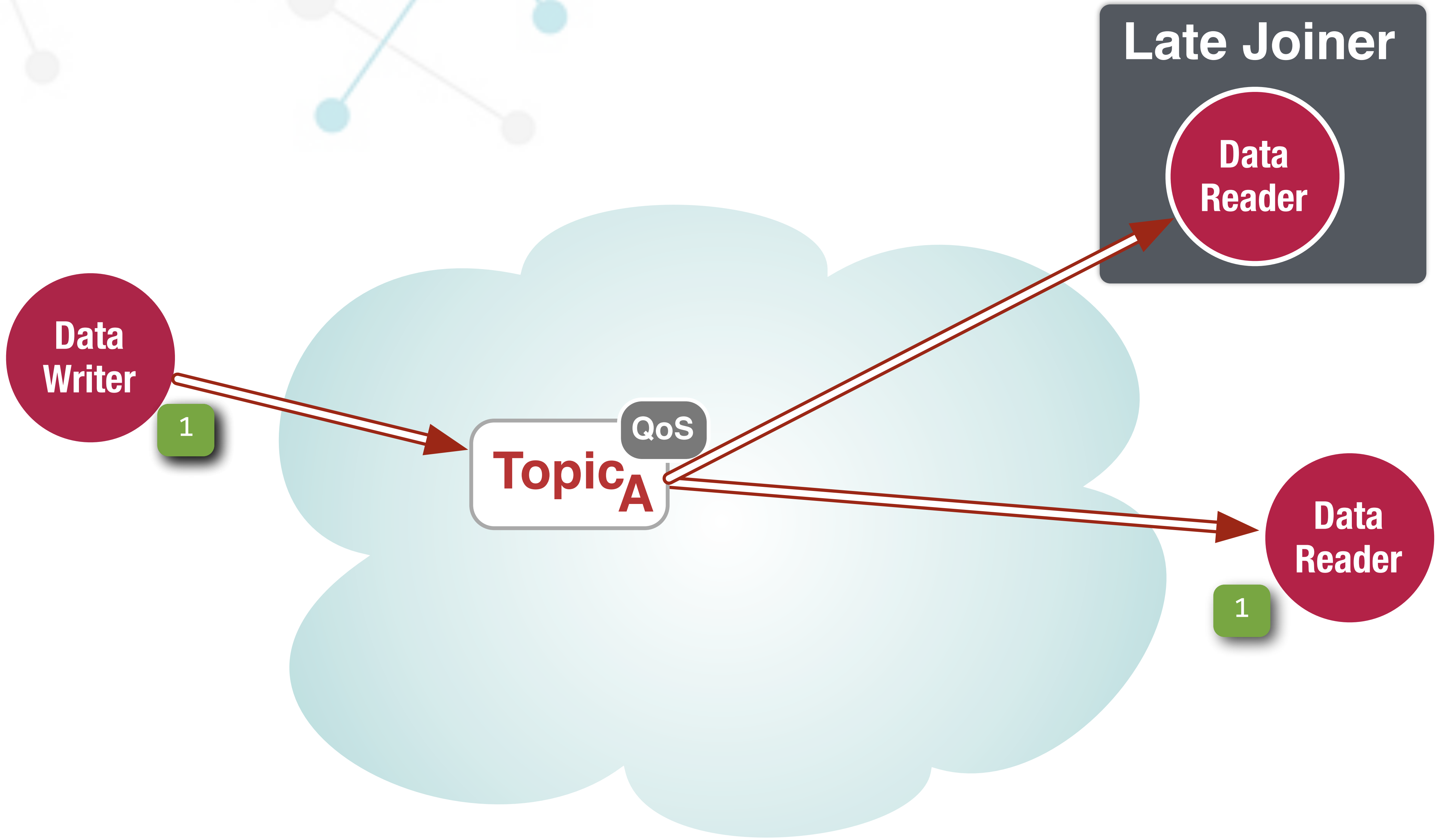


Some Time Decoupling

Data availability is tied to the availability of the data writer and the history settings

Copyright PrismTech, 2017

Transient Local

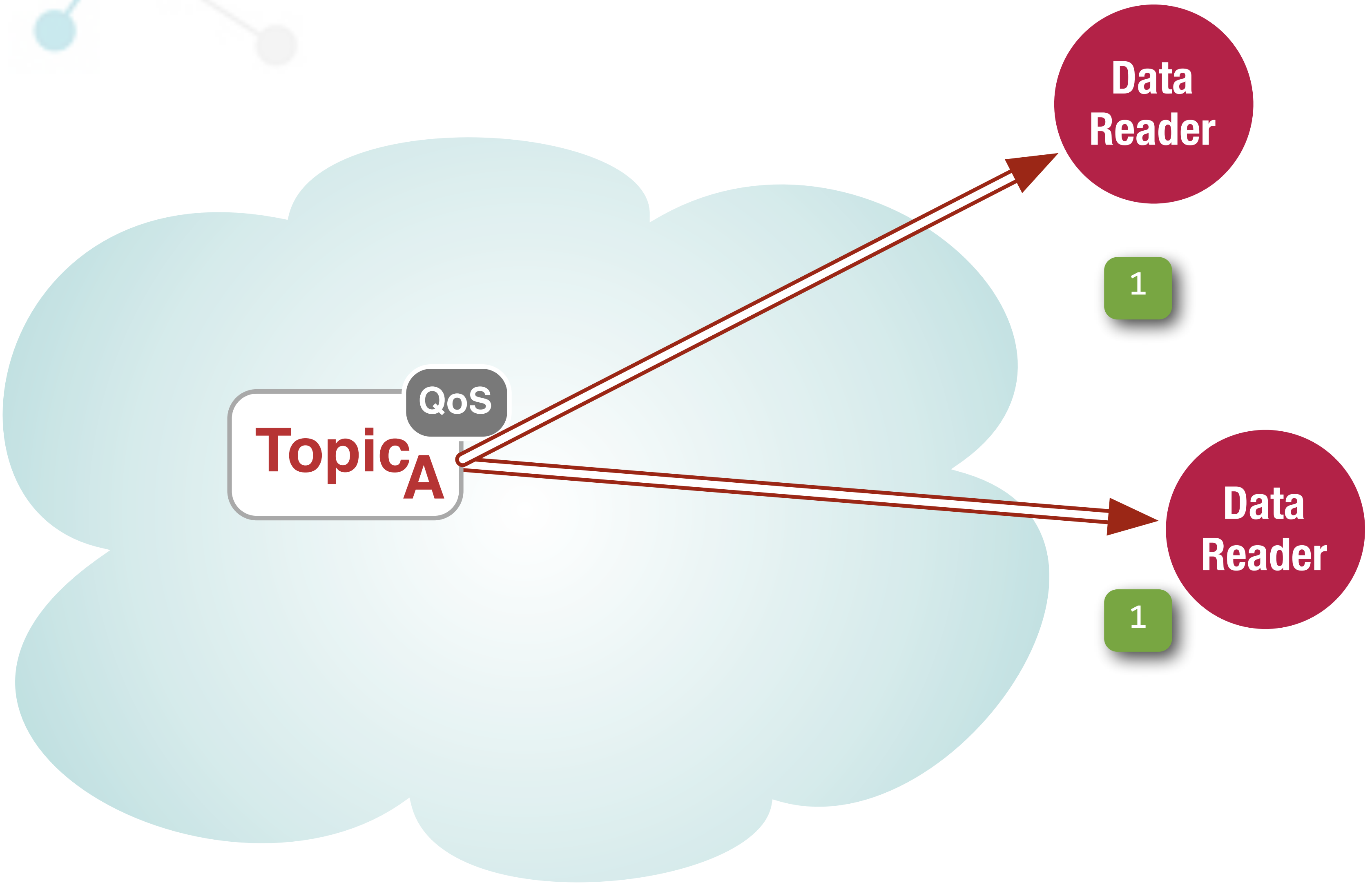


Some Time Decoupling

Data availability is tied to the availability of the data writer and the history settings

Copyright PrismTech, 2017

Transient Local

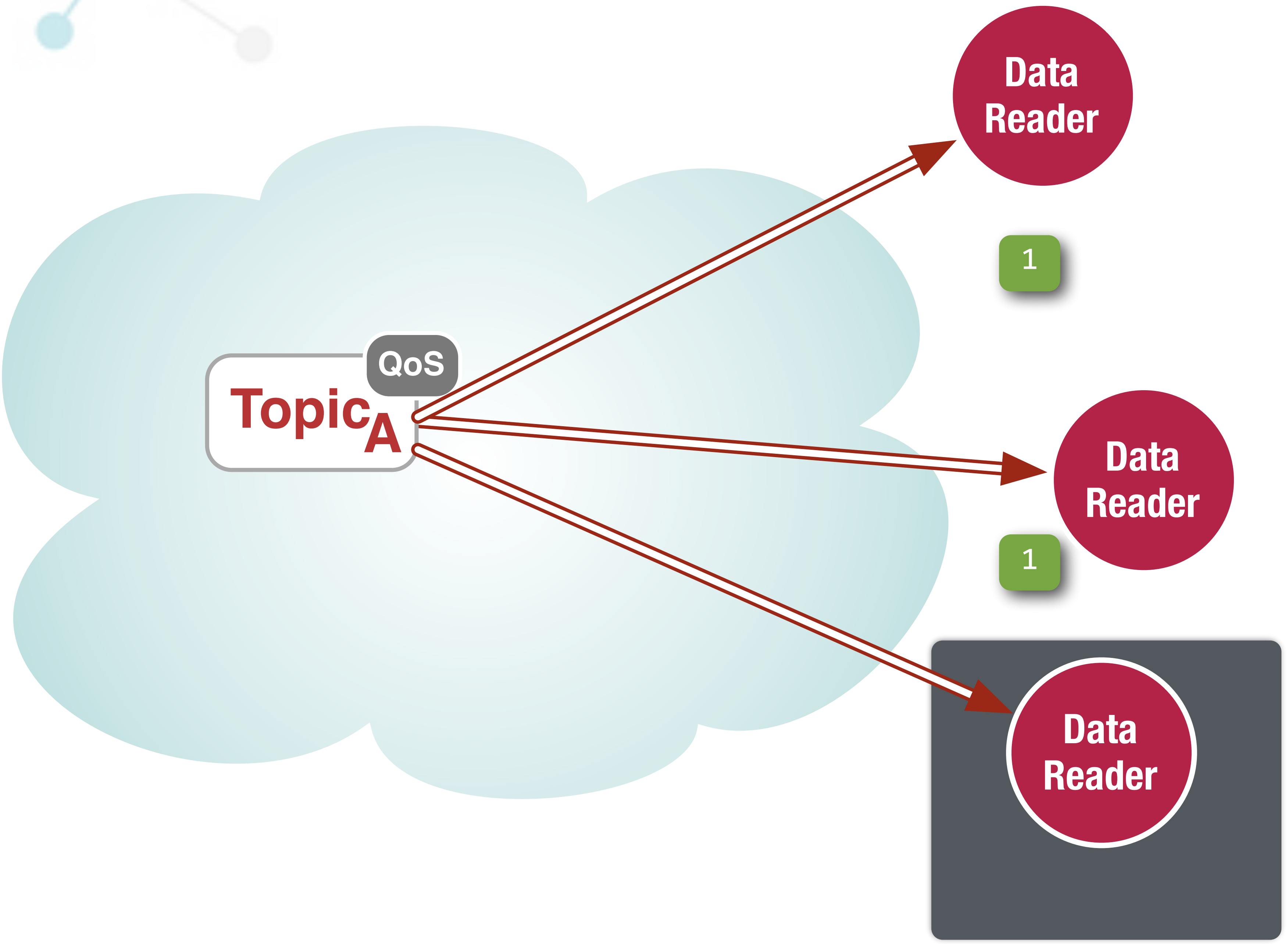


Some Time Decoupling

Data availability is tied to the availability of the data writer and the history settings

Copyright PrismTech, 2017

Transient Local



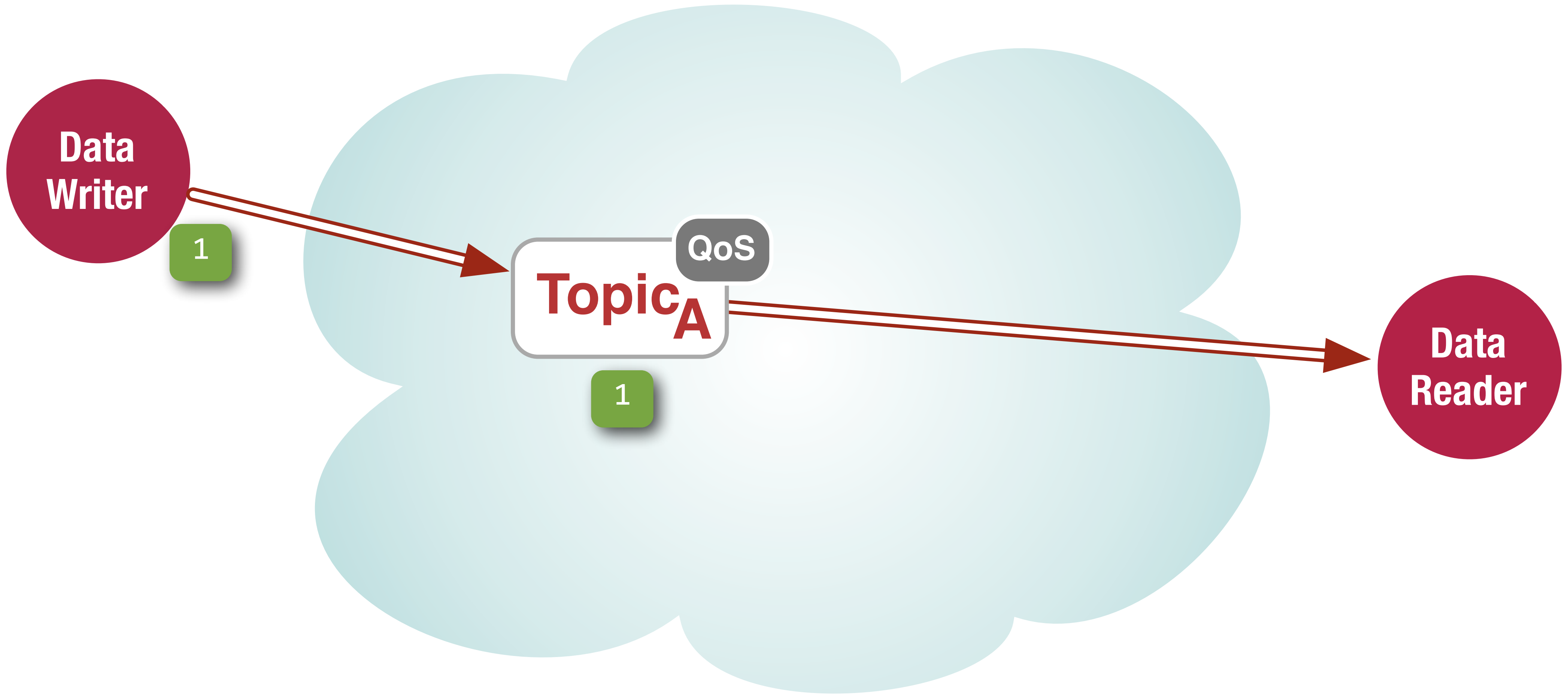
This late joiner will not get any data as the writer is gone…

Some Time Decoupling

Data availability is tied to the availability of the data writer and the history settings

Copyright PrismTech, 2017

Transient Durability

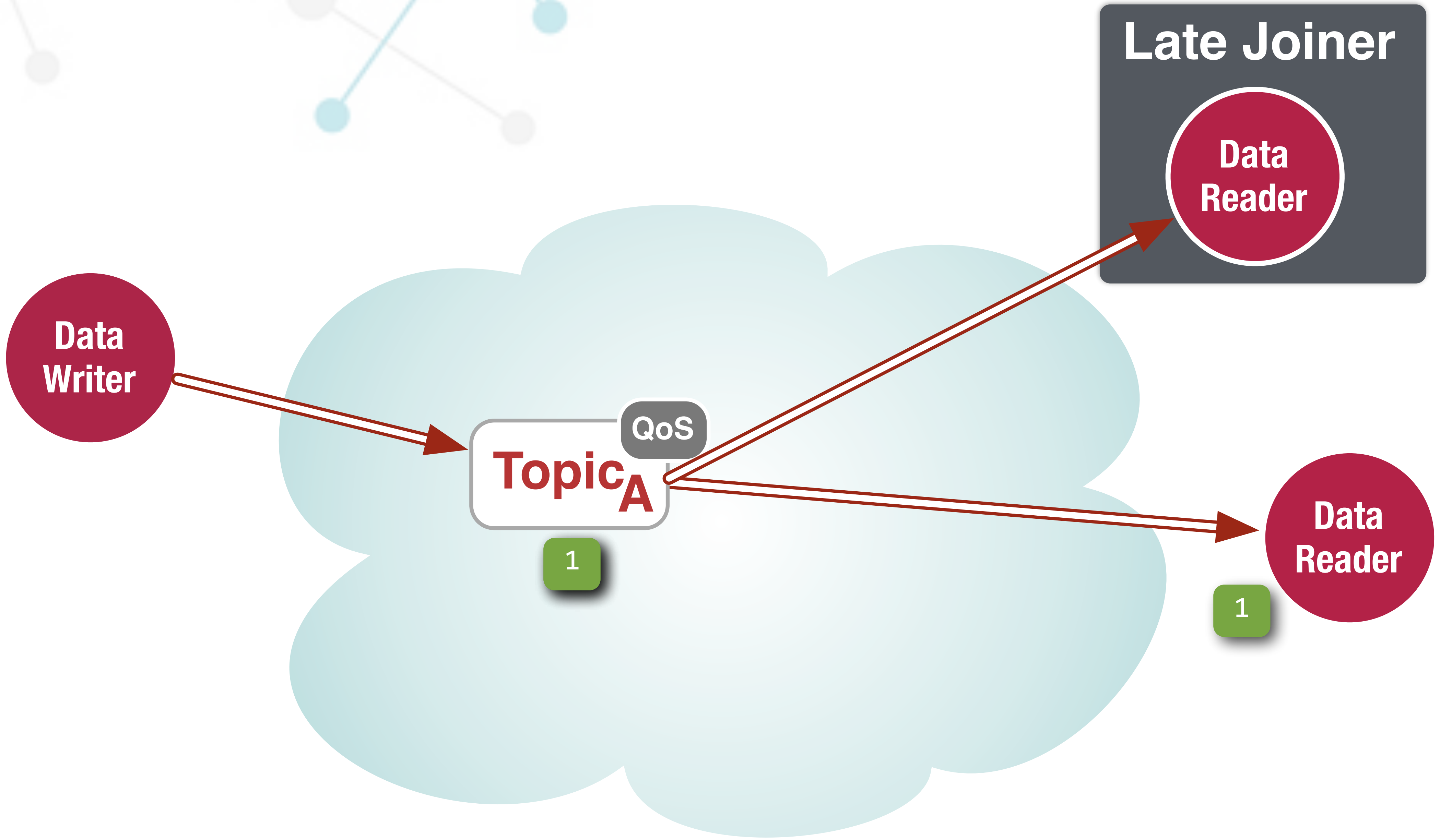


Time Decoupling

Data availability is tied to the availability of the durability service

Copyright PrismTech, 2017

Transient Durability

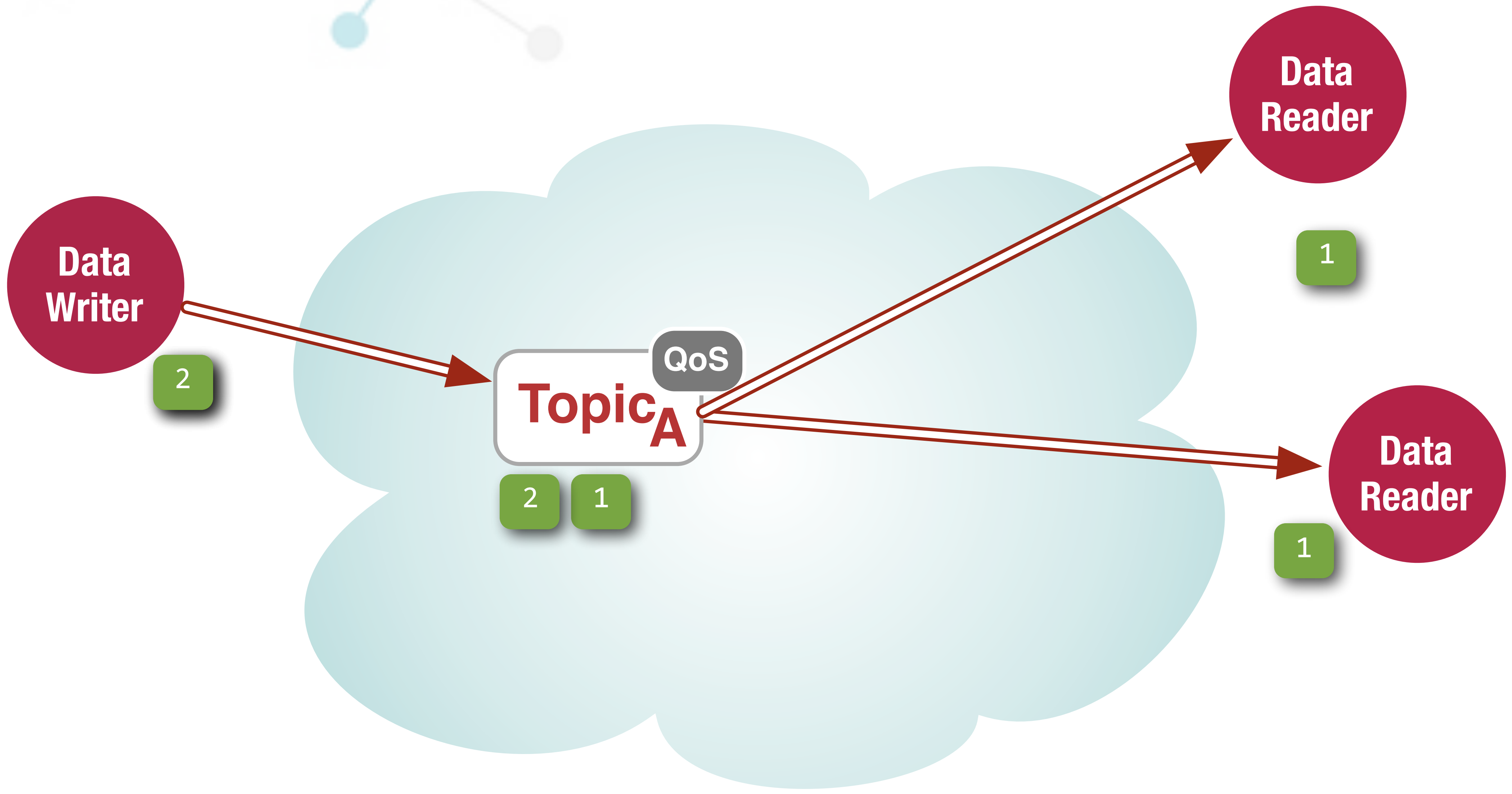


Time Decoupling

Data availability is tied to the availability of the durability service

Copyright PrismTech, 2017

Transient Durability

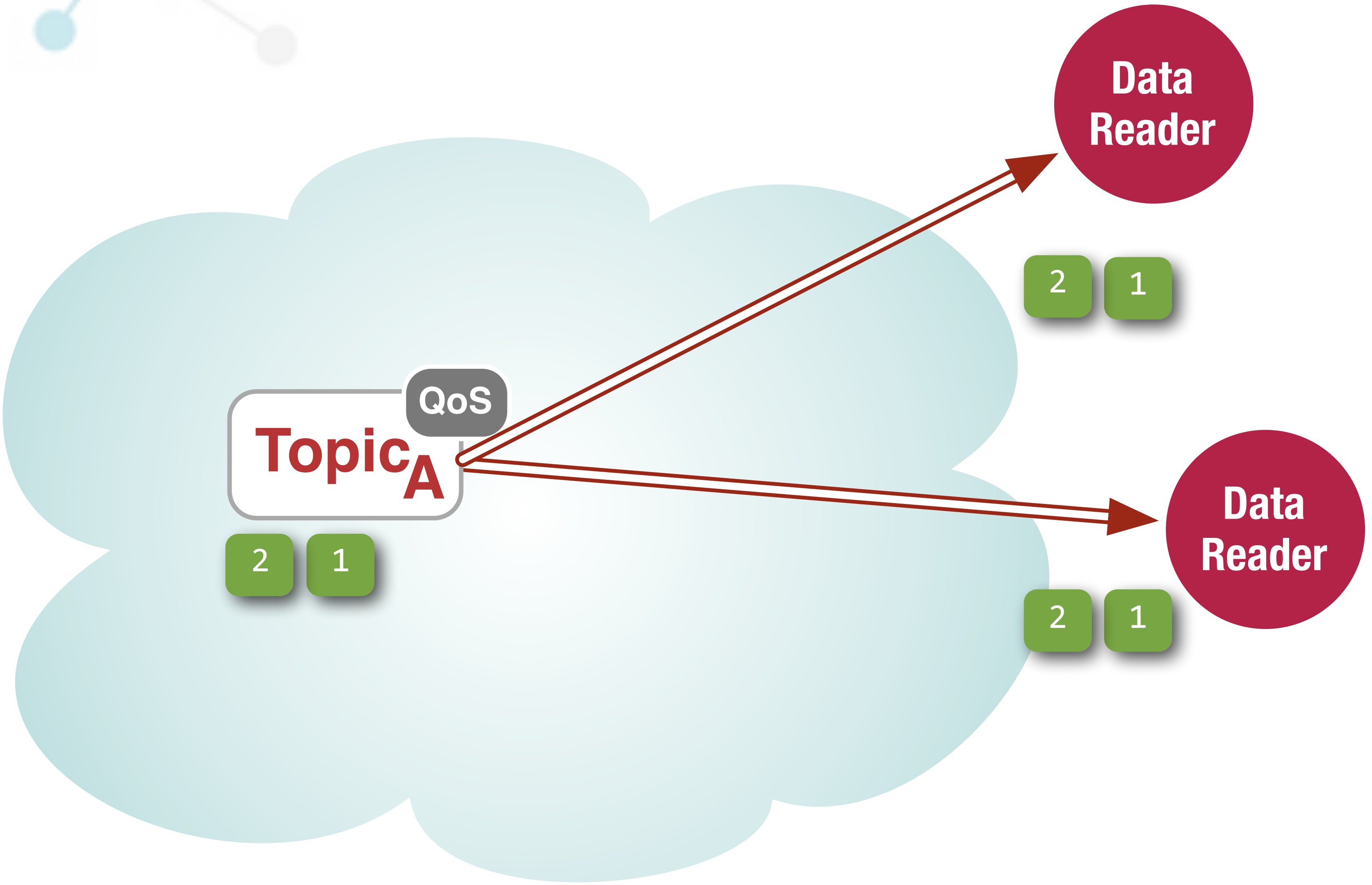


Time Decoupling

Data availability is tied to the availability of the durability service

Copyright PrismTech, 2017

Transient Durability

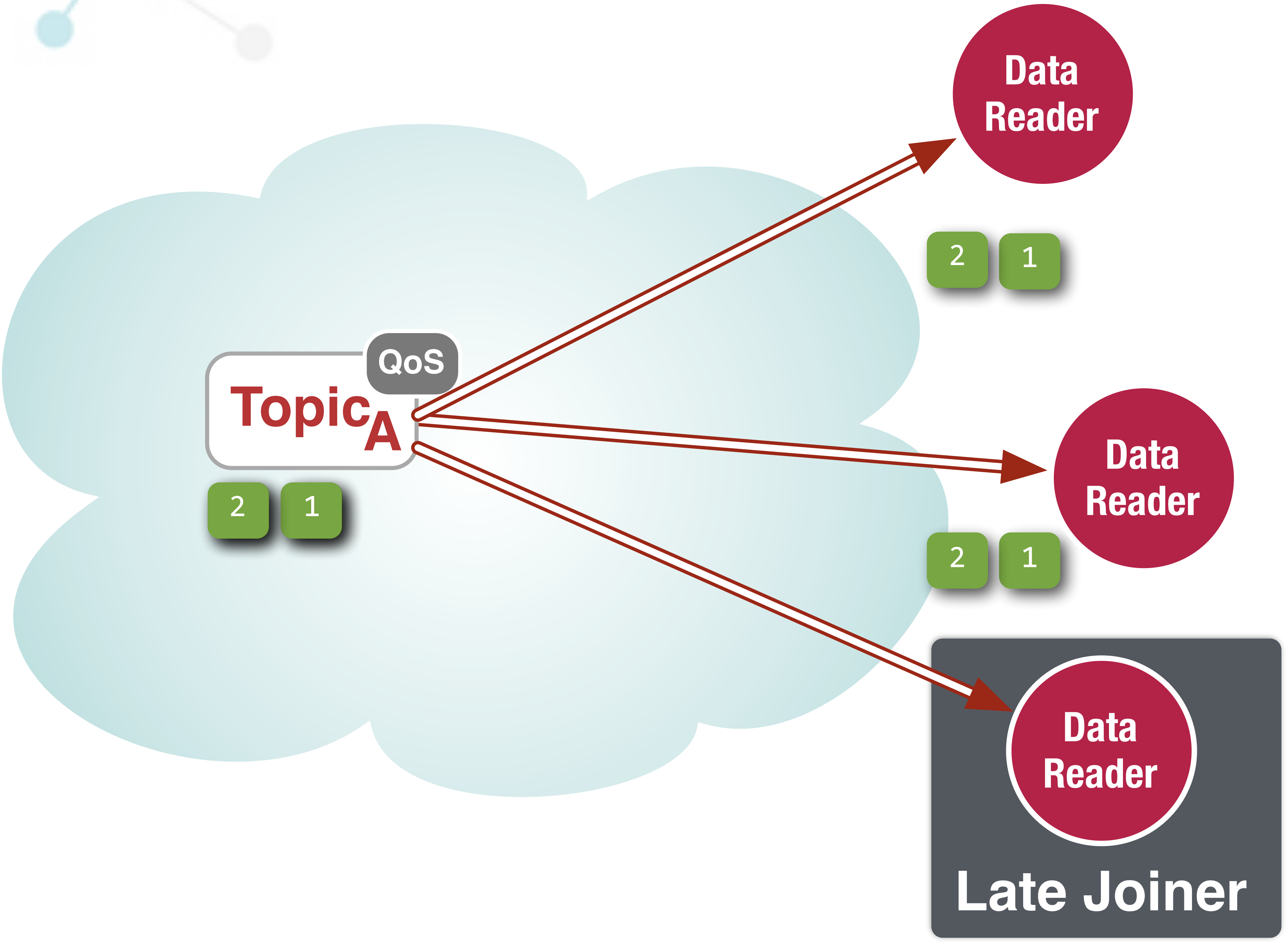


Time Decoupling

Data availability is tied to the availability of the durability service

Copyright PrismTech, 2017

Transient Durability



Time Decoupling

Data availability is tied to the availability of the durability service

Copyright PrismTech, 2017

Reliability

Copyright PrismTech, 2017

Data Reliability

The RELIABILITY QoS Policy controls one of the dimensions of reliability in DDS

Yet, the full semantics of DDS reliability is controlled by a combination of the RELIABILITY and the HISTORY QoS

QoS Policy	Applicability	RxO	Modifiable
RELIABILITY	T, DR, DW	Y	N

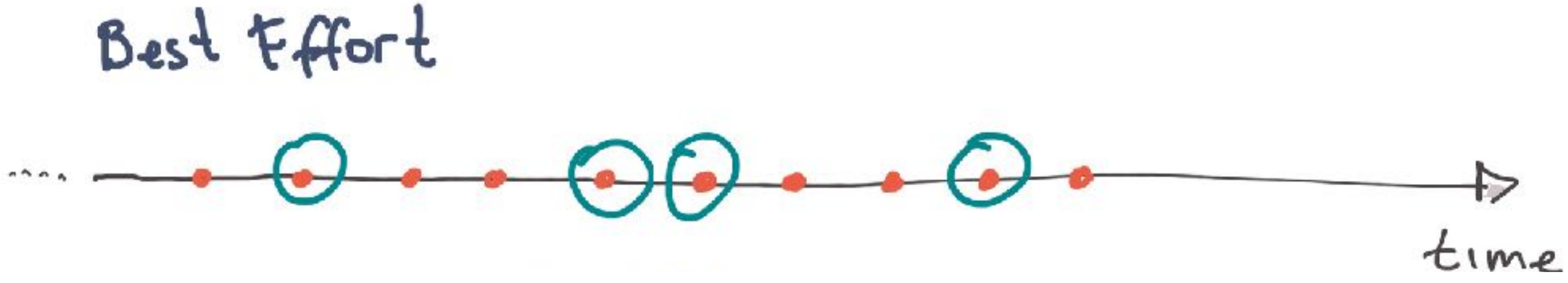
Copyright PrismTech, 2017

Best Effort

RELIBILITY = BEST EFFORT

DDS will deliver an arbitrary subsequence of the samples written against a Topic Instance

Samples may be dropped because of network loss or because of flow-control



Copyright PrismTech, 2017

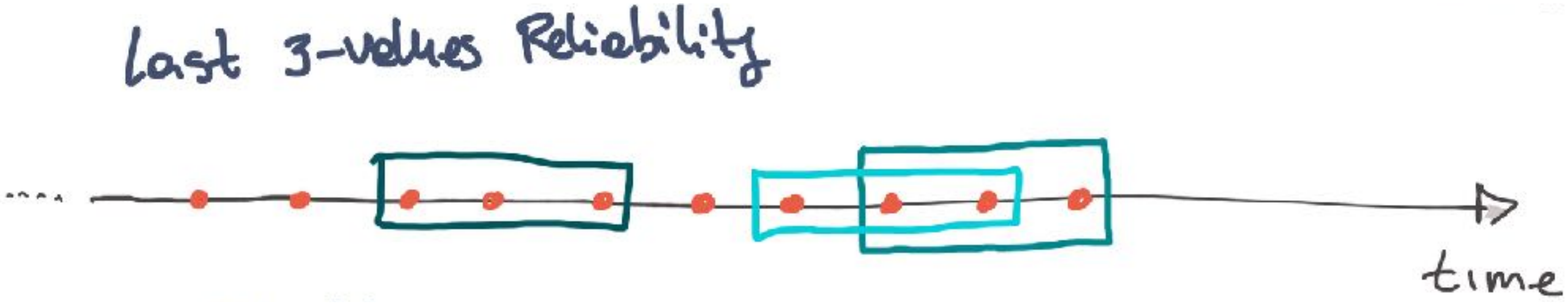
Last n-values Reliability

```
RELIABILITY = RELIABLE
HISTORY = KEEP_LAST(n)
```

Under stationary conditions an application is guaranteed to receive the last n-samples written for a Topic Instance

Samples falling outside the history may be dropped at the sending or receiving side for flow/resource control

Notice that this kind of reliability behaves as a circuit breaker for slow consumers

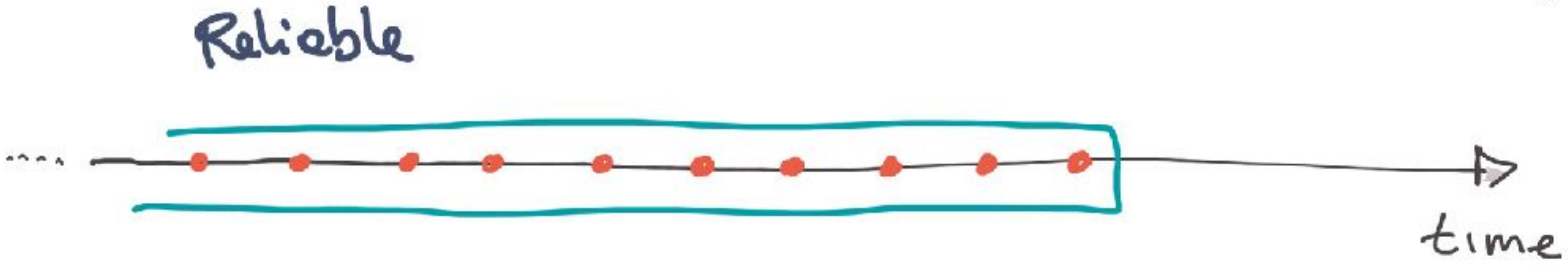


Copyright PrismTech, 2017

Reliable

```
RELIABILITY = RELIABLE
HISTORY = KEEP_ALL
```

All samples written against a Topic Instance are delivered. Since from a theoretical perspective reliability in asynchronous systems either violate progress or requires infinite memory, DDS provides QoS to control both resources as well as blocking time



Copyright PrismTech, 2017

Memory…

Copyright PrismTech, 2017

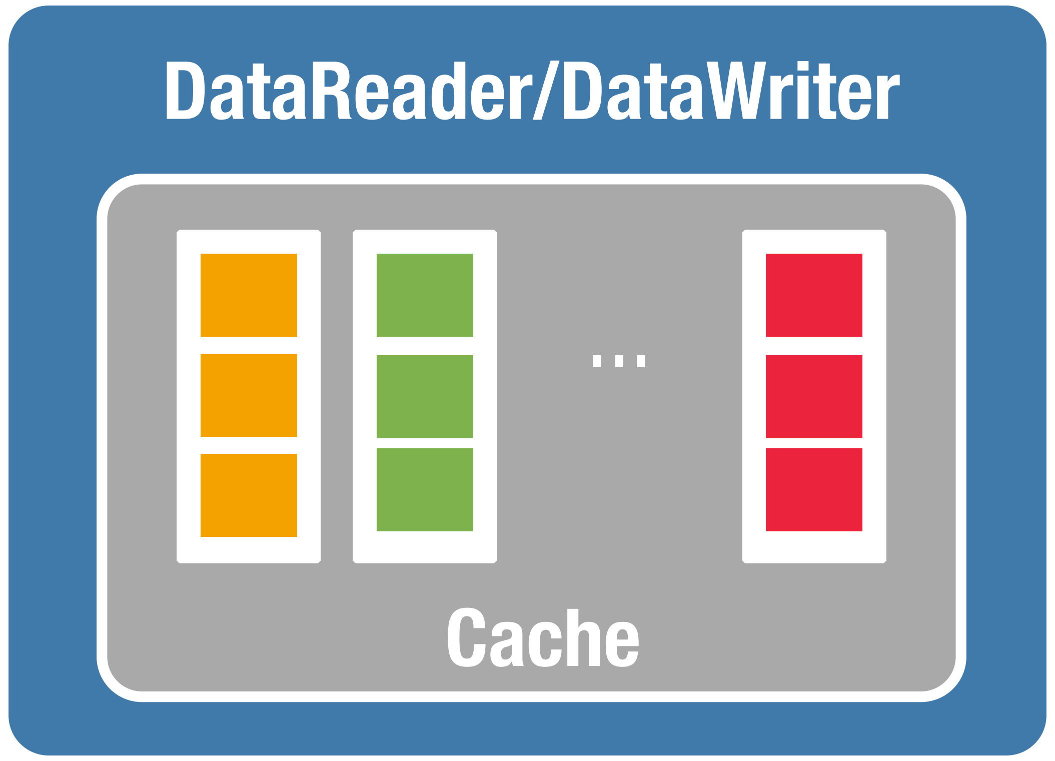
Resource Limits

The Resource Limits QoS Policy controls the maximum size of the DataReader and DataWriter cache through three parameters

`max_samples`: max number of samples the cache may hold

`max_instances`: max number of instances the cache may hold

`max_samples_ per_instance`: max number of samples allowed per instance



QoS Policy	Applicability	RxO	Modifiable
RESOURCE LIMITS	T, DR, DW	N	N

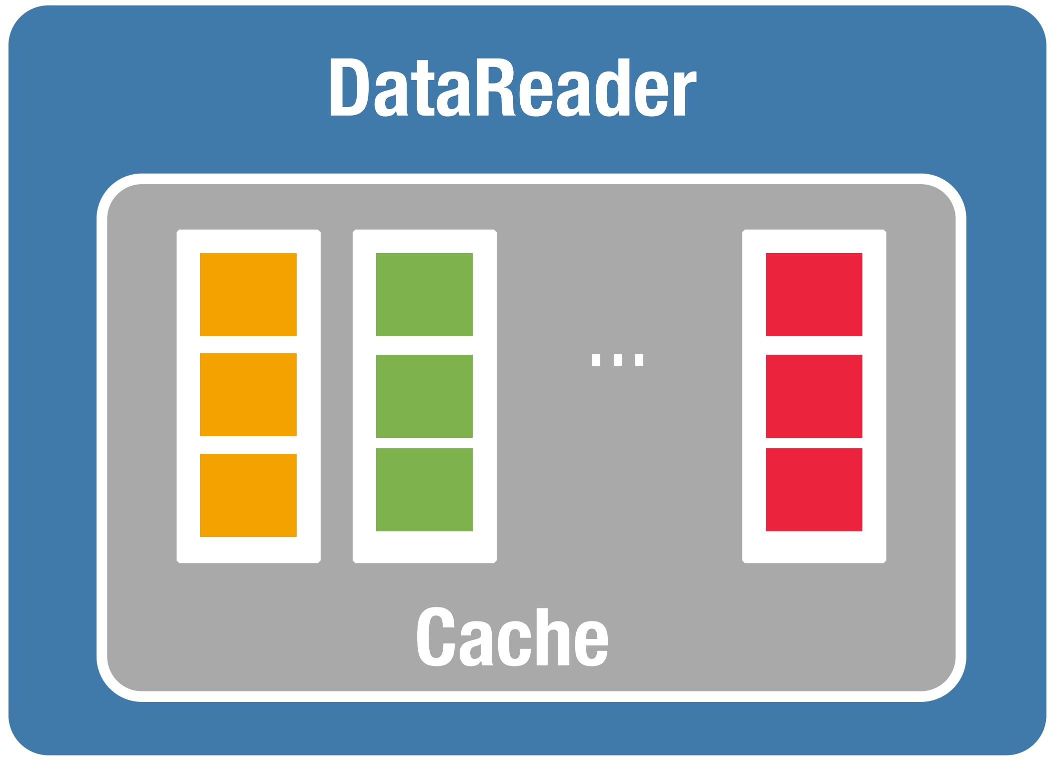
Copyright PrismTech, 2017

ReaderData Lifecycle

Configures the **purge delay** for instances that have **no writer or** which have been **disposed**

This is controlled through the `autopurge_nowriter_samples _delay` and the `autopurge_disposed_samples_delay` parameters

For both parameters the default delay is infinite



QoS Policy	Applicability	RxO	Modifiable
READER DATA LIFECYCLE	DR	N	Y

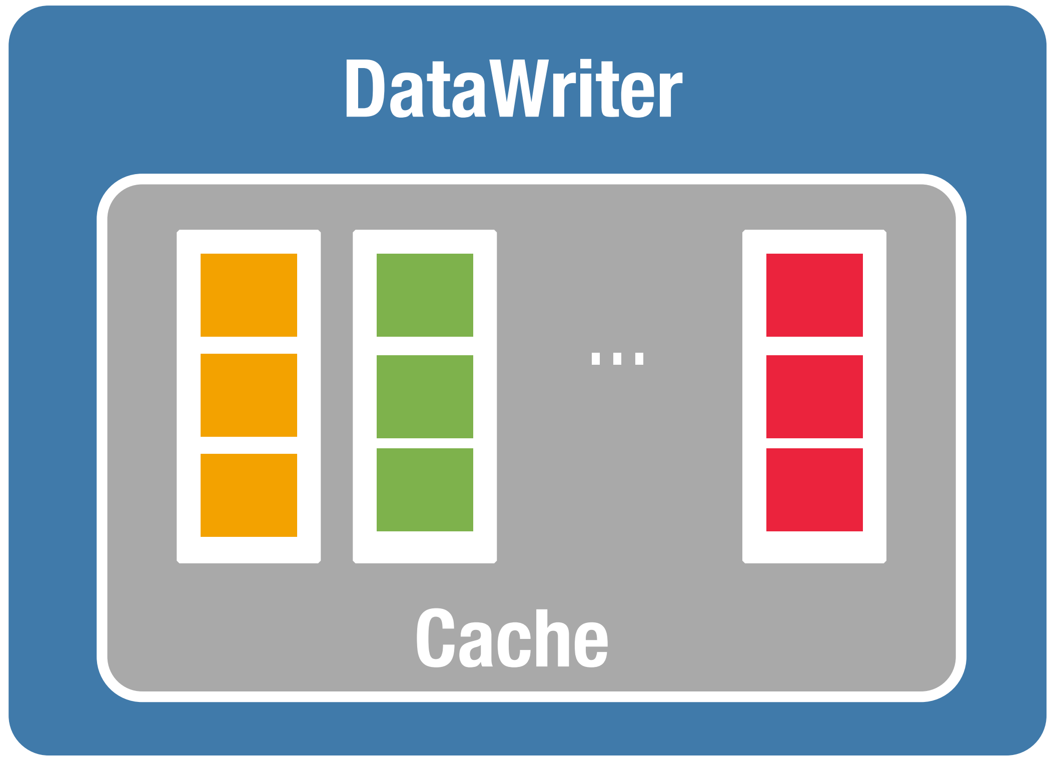
Copyright PrismTech, 2017

WriterData Lifecycle

Controls whether **unregistered instances** are **automatically disposed** or not

This is controlled through the `autodispose_unregistered_instance` parameter

This parameter is set to true by default



QoS Policy	Applicability	RxO	Modifiable
WRITER DATA LIFECYCLE	DW	N	Y

Copyright PrismTech, 2017

Fault-Tolerance

Copyright PrismTech, 2017

Ownership

Availability of data producers can be controlled via two QoS Policies:

OWNERSHIP (SHARED vs. EXCLUSIVE)

OWNERSHIP STRENGTH

Instances of exclusively owned Topics can be modified (are owned) by the higher strength writer

Writer strength is used to coordinate replicated writers

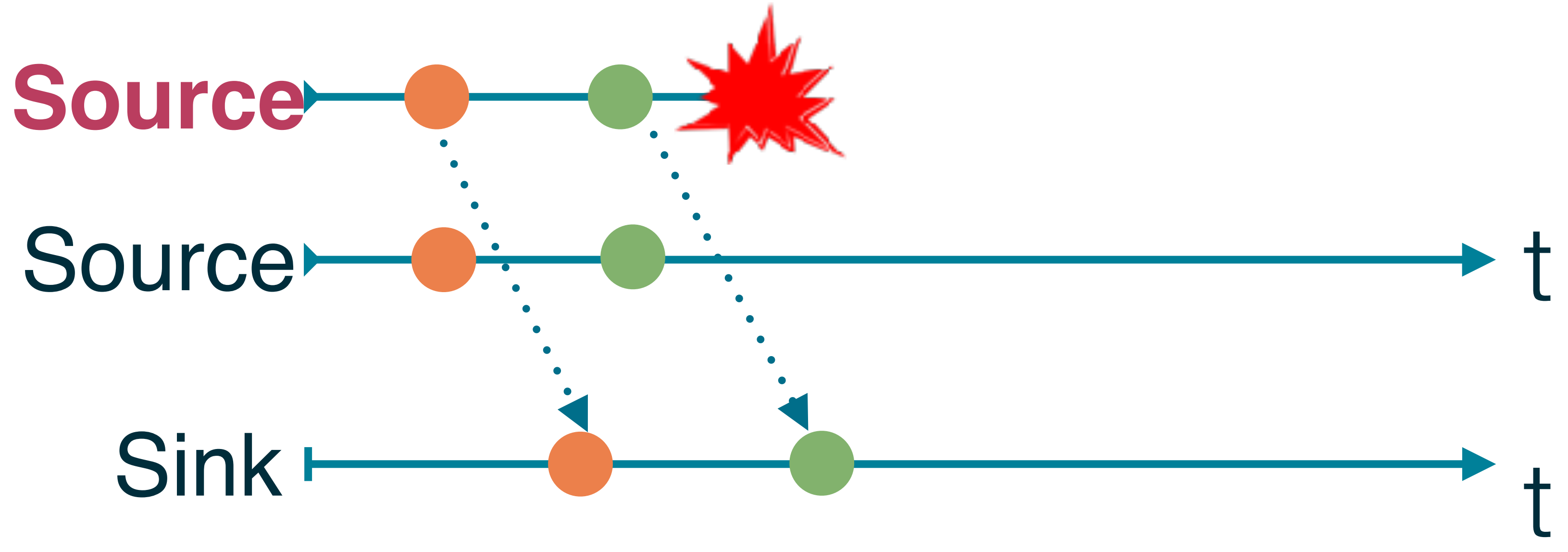
QoS Policy	Applicability	RxO	Modifiable
OWNERSHIP	T, DR, DW	Y	N
OWNERSHIP STRENGTH	DW	N	Y

Copyright PrismTech, 2017

Fault-Masking

The Ownership can be used as a fault-masking mechanism that allow to replicate **Sources** and transparently switch over when a failure occurs

At any point in time the "active" source is the one with the highest strength. Where the strength is an integer parameter controller by the user



Copyright PrismTech, 2017

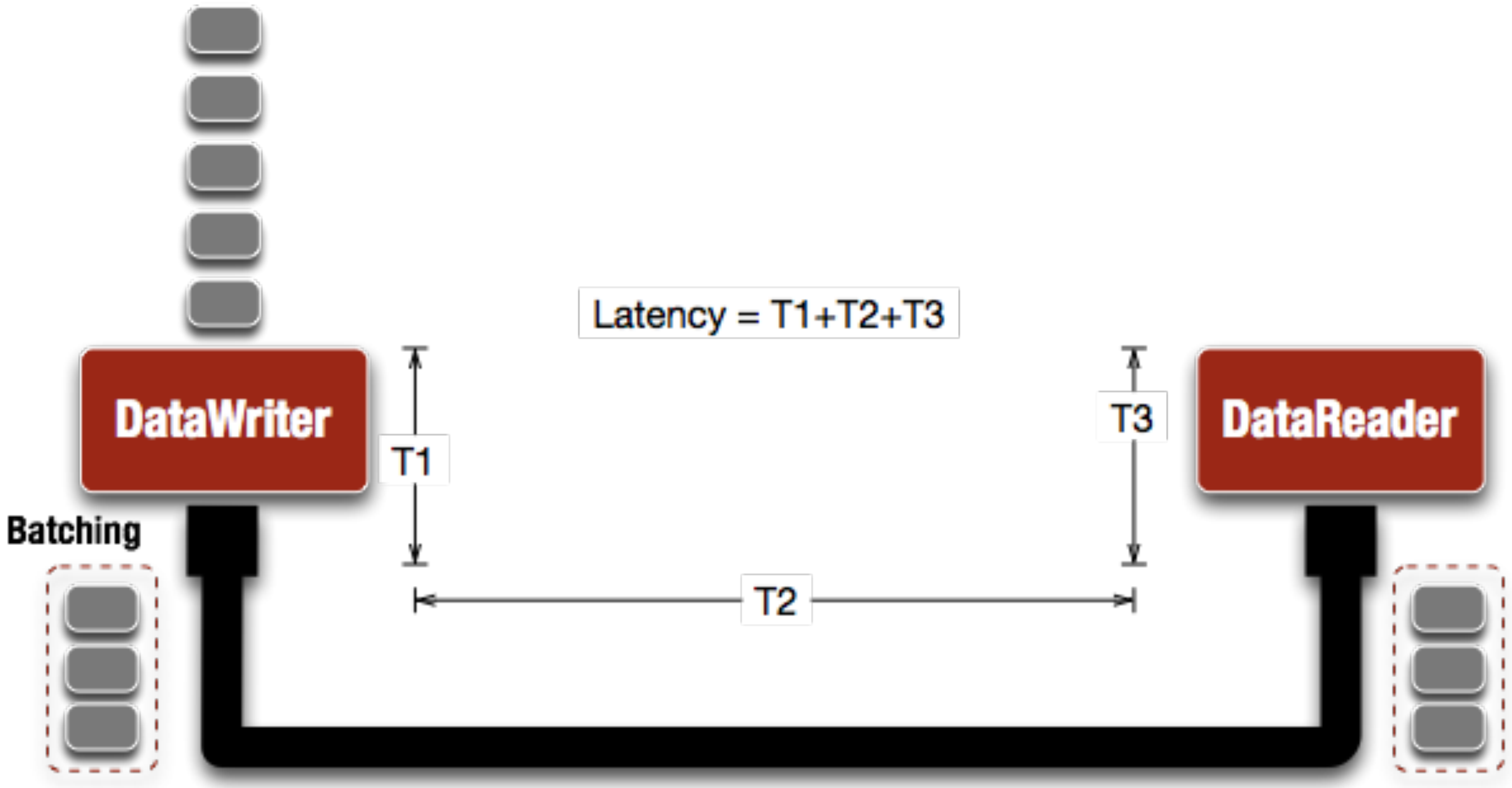
Temporal Properties

Copyright PrismTech, 2017

Latency Budget

The LATENCY_BUDGET QoS policy specifies the maximum acceptable delay from the time the data is written until the data is inserted in the receiver's application-cache

A non-zero latency-budget allows a DDS implementation to batch samples and improve CPU/Network utilisation



QoS Policy	Applicability	RxO	Modifiable
LATENCY BUDGET	T, DR, DW	Y	Y

Copyright PrismTech, 2017

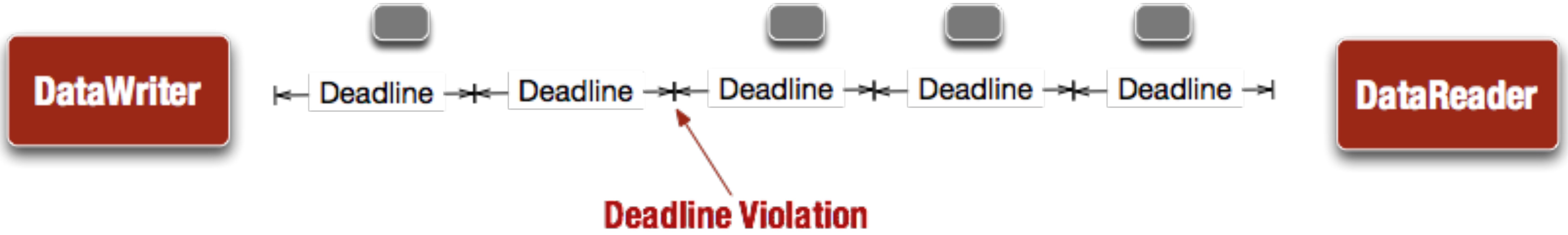
Deadline

QoS Policy	Applicability	RxO	Modifiable
DEADLINE	T, DR, DW	Y	Y

The DEADLINE QoS policy defines the maximum inter-arrival time between data samples

DataWriter indicates that the application commits to write a new sample at least once every deadline period

DataReaders are notified when the DEADLINE is violated



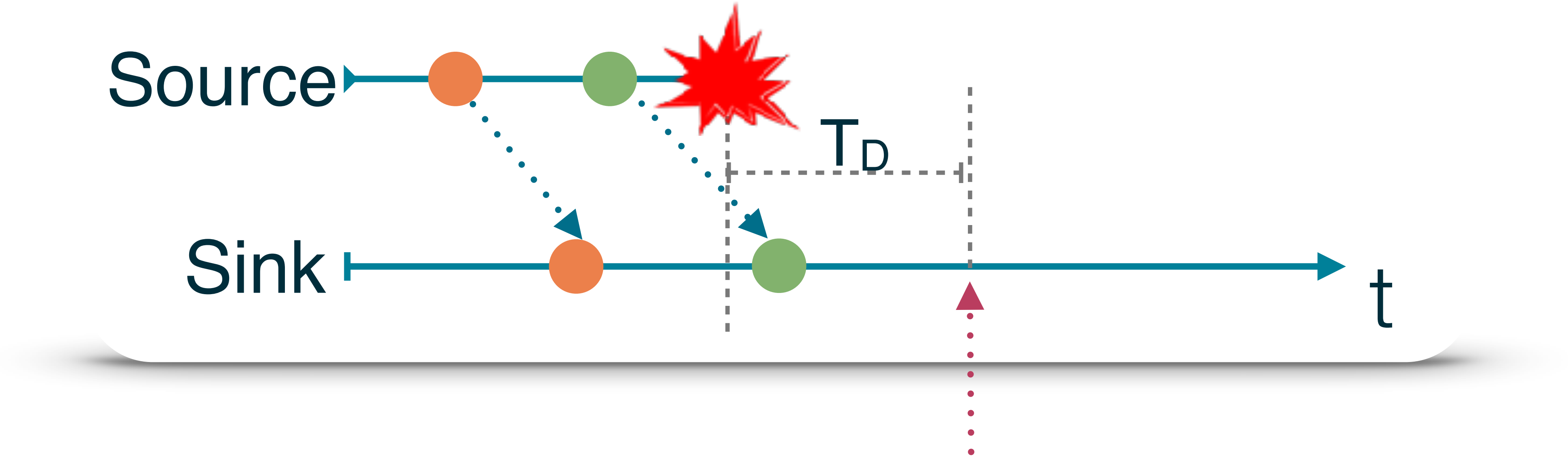
Copyright PrismTech, 2017

Failure Detection

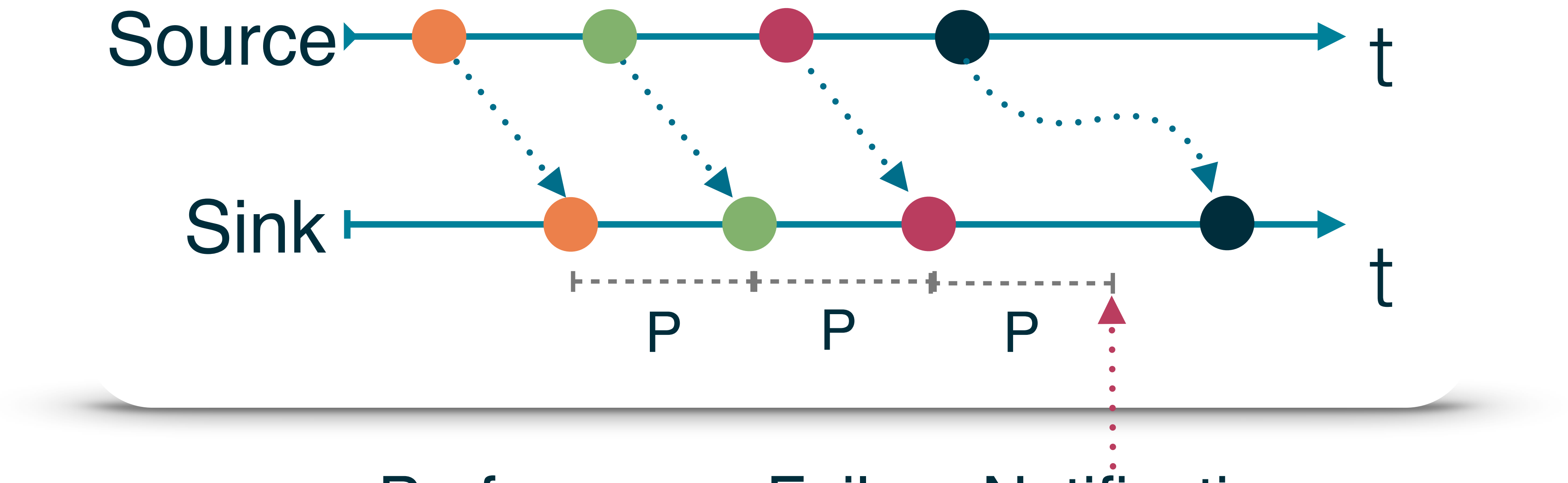
DDS provides mechanism for detecting traditional faults as well as performance failures

The Fault-Detection mechanism is controlled by means of the DDS Liveliness policy

Performance Failures can be detected using the Deadline Policy which allows to receive notification when data is not received within the expected delays



Fault Notification



Performance Failure Notification

Copyright PrismTech, 2017

Transport Priority

The TRANSPORT_PRIORITY QoS policy is a **hint** to the infrastructure as to how to set the **priority** of the **underlying transport** used to **send** the **data**.

QoS Policy	Applicability	RxO	Modifiable
TRANSPORT PRIORITY	T, DW	N	Y

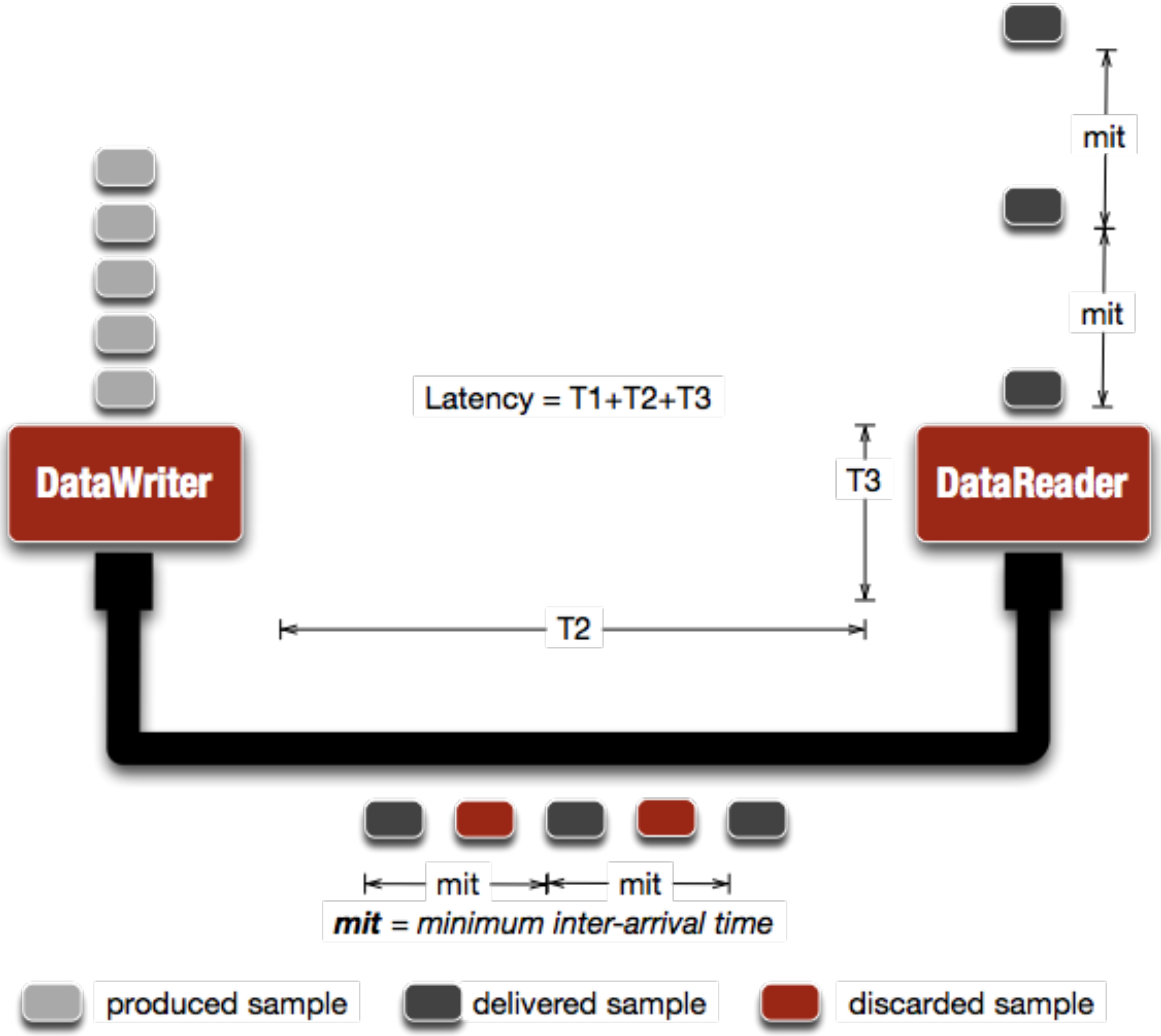
Copyright PrismTech, 2017

QoS Policy	Applicability	RxO	Modifiable
TIME BASED FILTER	DR	N	Y

Time-Based Filter

The Time Based Filter allows to control the throughput at which data is received by a data reader

Samples produced more often than the minimum inter-arrival time are not delivered to the data reader



mit *= minimum inter-arrival time*

Copyright PrismTech, 2017

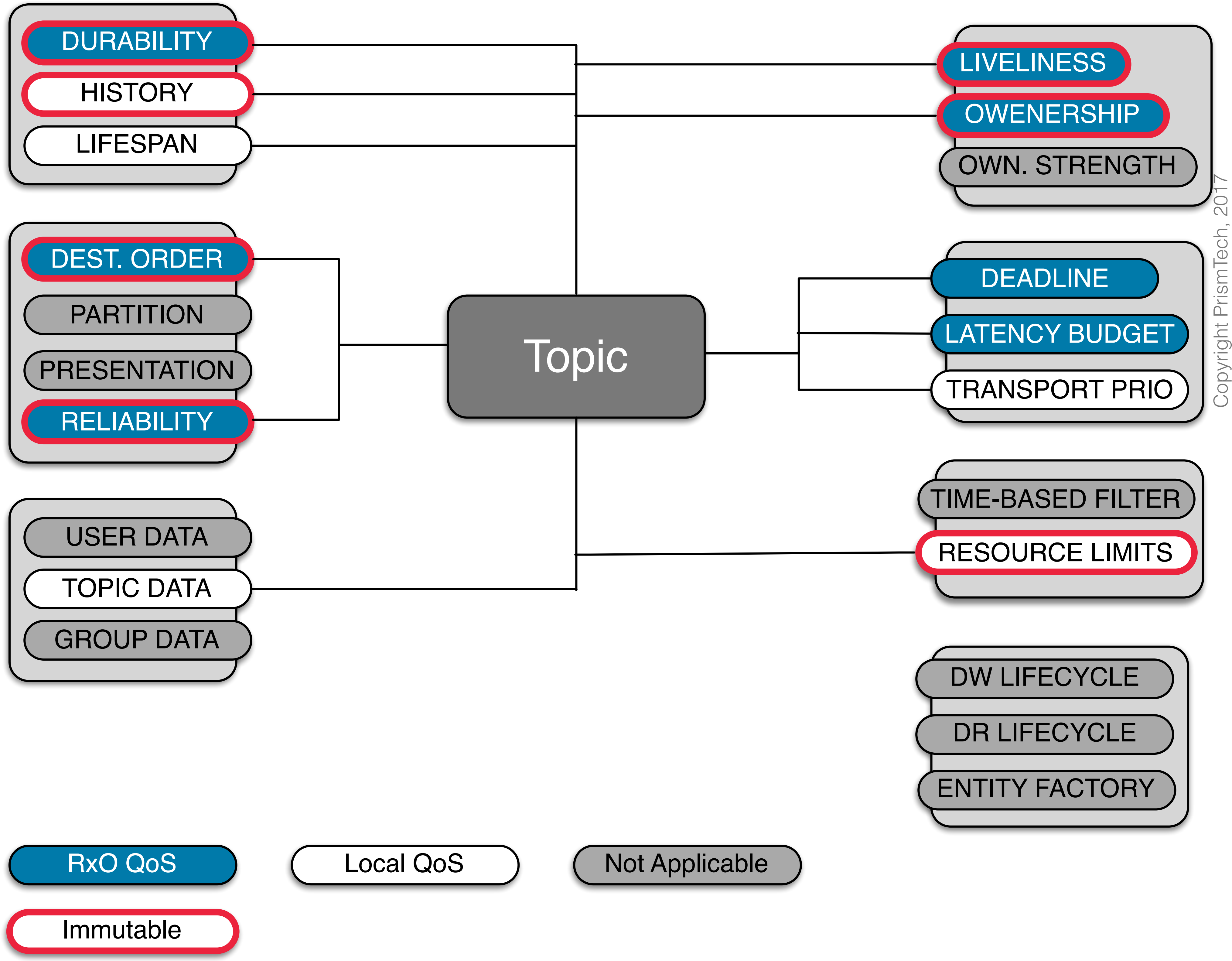
QoS Modeling Idioms

System-Level QoS Policies

Identify the QoS Policies that capture the key non-functional properties of the various kinds of information flow in your system

Define these QoS as Topics QoS

Create DataReaders and DataWriters but inheriting the Topic QoS



Copyright PrismTech, 2017

QoS Provider

Allo for deploy-time refinement of DDS QoS Policies

To allow for this leverage the QoS Provider

```
// QosProvider...
QosProvider qos_provider(
      “file:///some/meaningful/path/qos.xml”,
      “my-qos-profile");

DataReader<AType> dr(sub, topic, qos_provider.datareader_qos());
```

Copyright PrismTech, 2017

State vs. Events

DDS provides first class support for modelling distributed state and events

Different QoS combination should be used to associate with the topic representing a state or an event the proper semantics

Copyright PrismTech, 2017

Soft State

In distributed systems you often need to model **soft-state -- a state that is periodically updated**

Examples are the reading of a sensor (e.g. Temperature Sensor), the position of a vehicle, etc.

The QoS combination to model **Soft-State** is the following:

```
Reliability        =>   BestEffort
Durability         =>   Volatile
History            =>   KeepLast(n) [with n = 1 in most of the cases]
Deadline           =>   updatePeriod
LatencyBudget      =>   updatePeriod/3 [rule of thumb]
DestinationOrder =>   SourceTimestamp [if multiple writers per instance]
```

Copyright PrismTech, 2017

Hard State

In distributed systems you often need to model **hard-state** -- a **state** that is **sporadically updated** and that often has **temporal persistence requirements**

Examples are system configuration, a price estimate, etc.

The QoS combination to model Hard-State is the following:

```
Reliability         =>  Reliable
Durability          =>  Transient | Persistent
History             =>  KeepLast(n) [with n = 1 in most of the cases]
DestinationOrder    =>  SourceTimestamp [if multiple writers per instance]
WriterDataLifecycle => autodispose_unregistered_instances = false
```

Copyright PrismTech, 2017

Event

In distributed systems you often need to model **events** -- the **occurrence of something noteworthy for our system**

Examples are a collision alert, the temperature beyond a given threshold, etc.

The **QoS** combination to model Events is the following:

```
Reliability       =>  Reliable
Durability        =>  any        [depends on system requirements]
History           =>  KeepAll [on both DataWriter and DataReader!]
DestinationOrder  =>  SourceTimestamp
WriterDataLifecycle => autodispose_unregistered_instances = false
ResourceLimits => [define appropriate bounds]
```

Copyright PrismTech, 2017

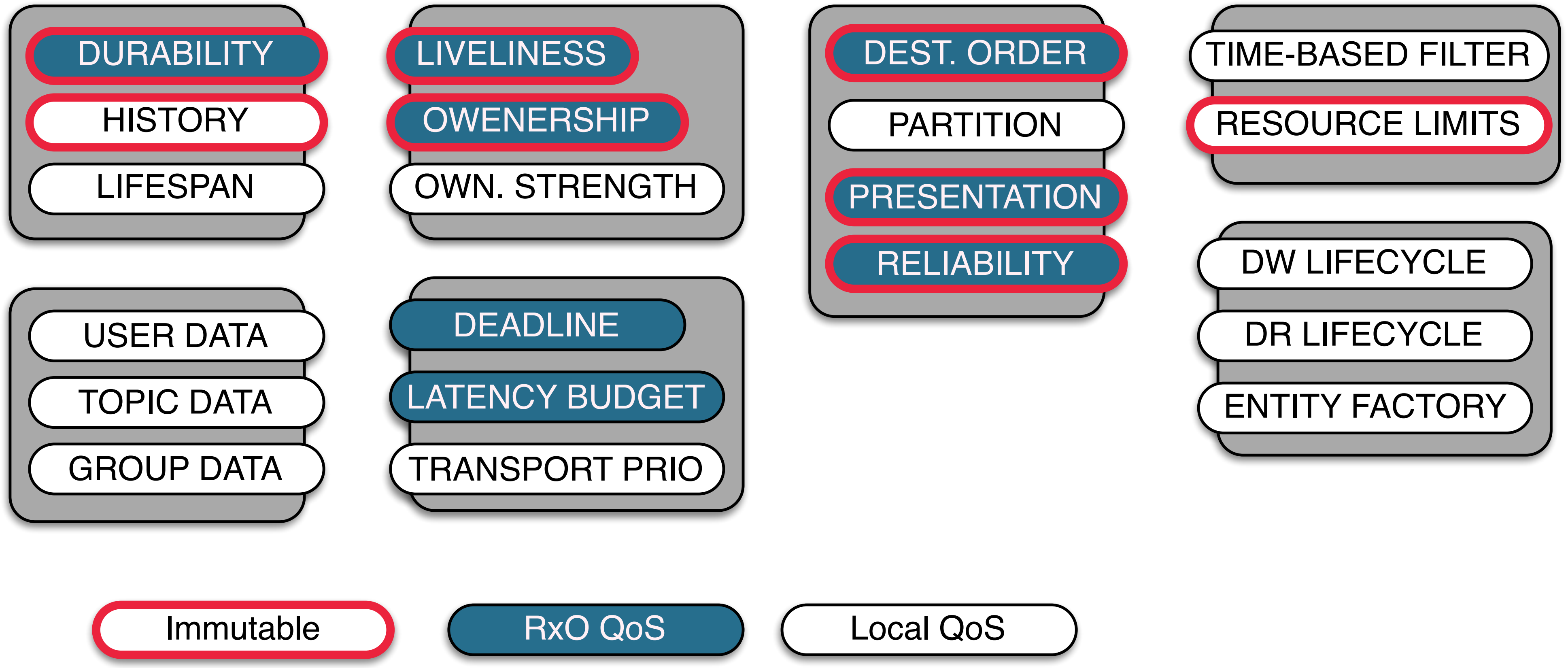
Summing Up

Copyright PrismTech, 2017

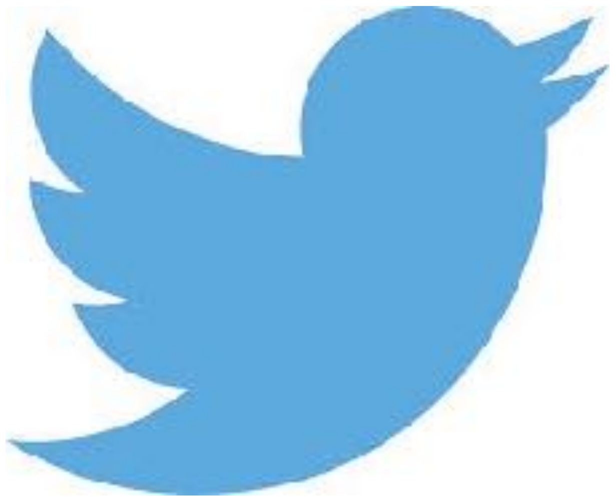
Final Remarks

DDS provides a rich set of QoS Policies to control the key aspects of data distribution, availability and resource utilisation

These QoS Policies are often applied in synergies to implement key patters



Copyright PrismTech, 2017



Copyright PrismTech, 2017