Angelo Corsaro, Ph.D.
Chief Technology Officer
OMG DDS SIG Co-Chair
angelo.corsaro@prismtech.com
PRISMTECH
Powering Netcentricity

DDS vs. AMQP

Agenda

- Genesis
- Technology Comparison
- Code Examples
- Concluding Remarks

OpenSplice | DDS

Delivering Performance, Openness, and Freedom

Historical Perspectives

Genesis

DDS

- Emerged from Aerospace and Defense to address the data distribution requirement of a large class of mission-critical systems
- Evolved to address on-the-wire interoperability and provide the ubiquitous data-bus for mission-critical System-of-Systems

AMQP

- Emerged from the Financial Market from the desire of freeing users from proprietary and non-interoperable messaging systems
- Evolved into an effort to define a generic enterprise messaging standard

Standardization Organization

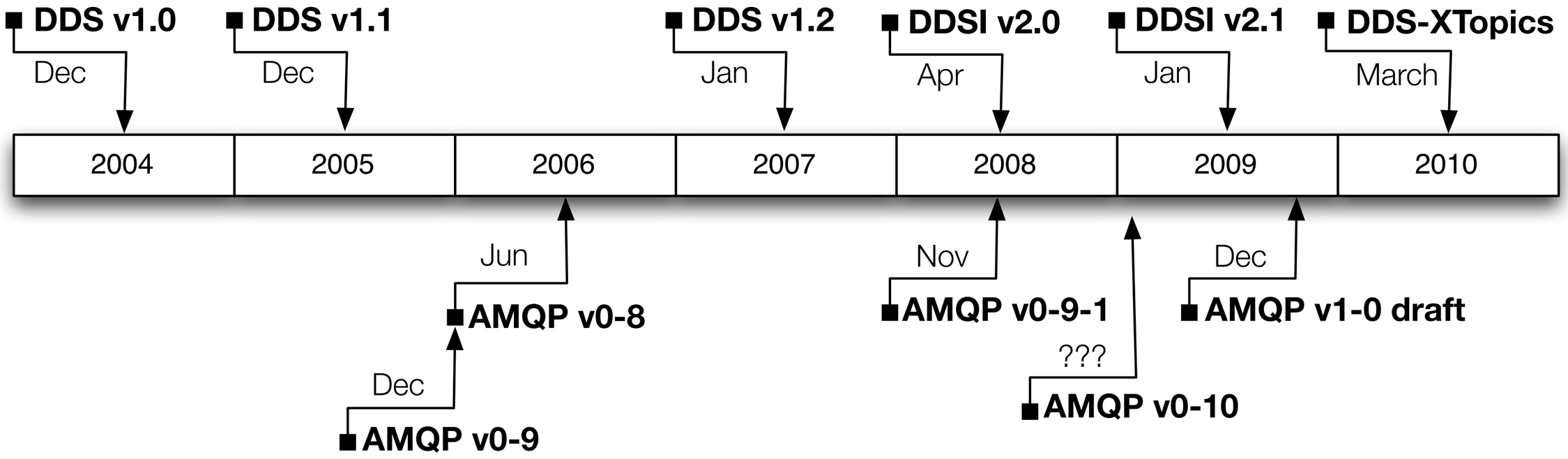
DDS

- DDS is an Object Management Group (OMG) Standard
- The OMG is a an international, open membership, not-for-profit computer industry consortium since 1989
- OMG is an ISO PAS submitter, able to submit our specifications directly into ISO's fast-track adoption process.

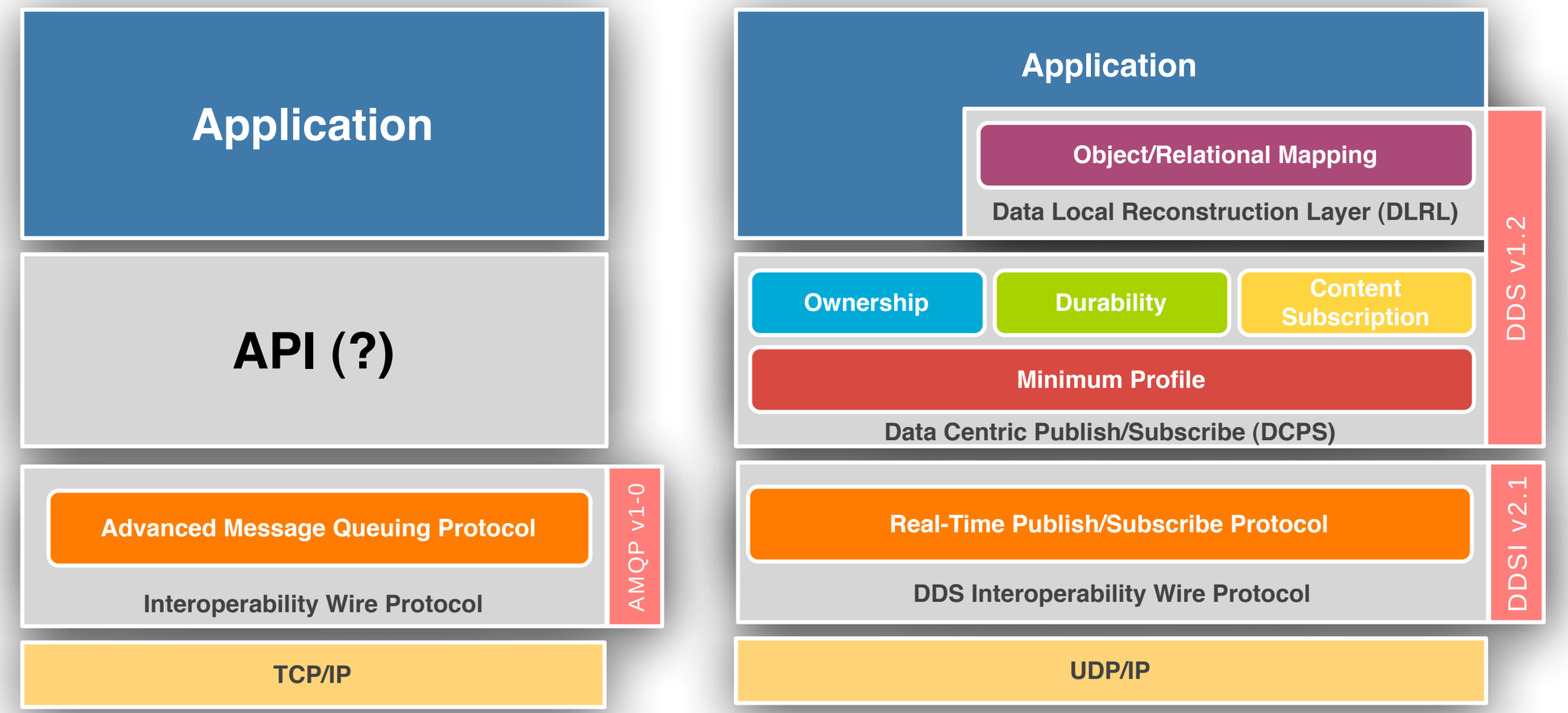
AMQP

- AMQP is standardized by the AMQP Working Group
- The AMQP Working Group is a non-profit organization with a free membership based on interest and merit

Standard Evolution



Scope of Standardization



OpenSplice | DDS

Delivering Performance, Openness, and Freedom

Technology Comparison

Which Versions?

In the reminder of this presentation I'll compare DDS v1.2 / DDSI v2.1 with AMQP v1-0

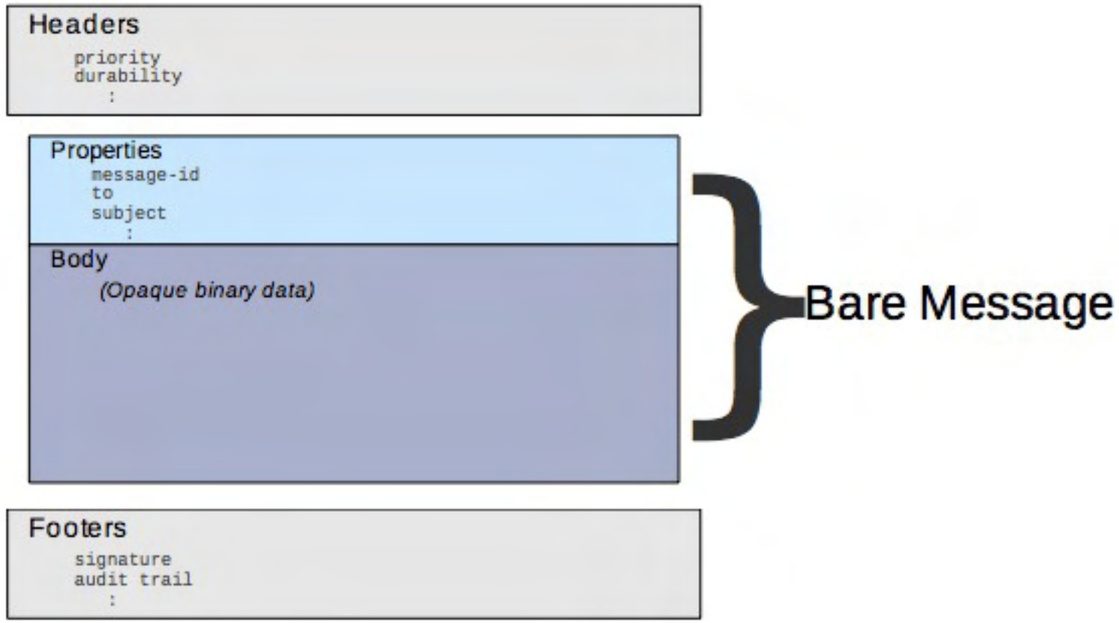
OpenSplice | DDS

Delivering Performance, Openness, and Freedom

Messages vs. Topics

AMQP v1-0 Messages

- AMQP is a standard protocol for messaging
- As such AMQP the unit of information that can be sent or received is a message.
- An AMQP message encapsulate the "Bare Message", provided by the application, within an annotated message

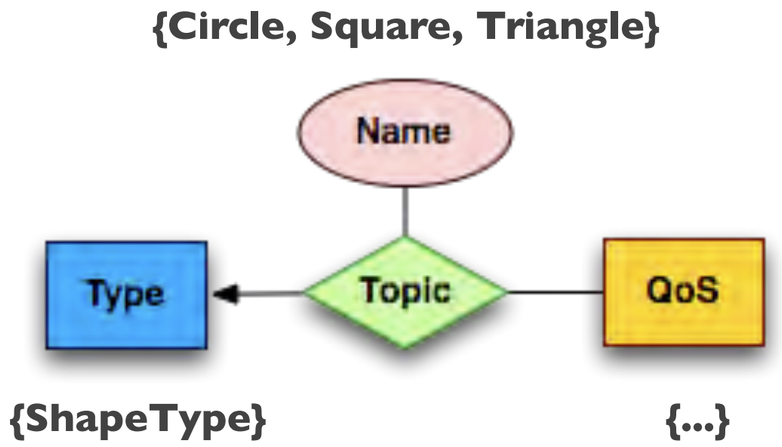


[Source AMQP specification v1-0]

DDS Topics

Topic

- Unit of information exchanged between Publisher and Subscribers.
- An association between a unique name, a type and a QoS setting



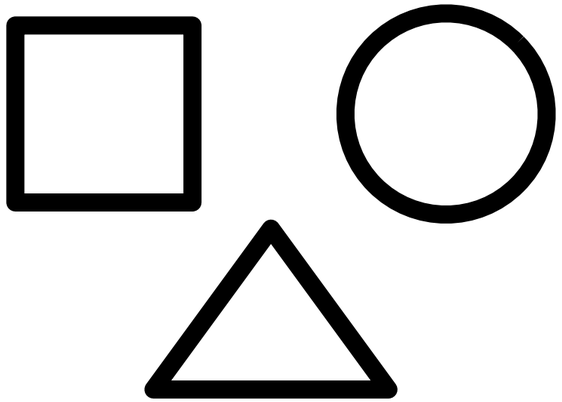
Topic Type.

- Type describing the data associated with one or more Topics
- A Topic type can have a key represented by an arbitrary number of attributes
- Expressed in IDL

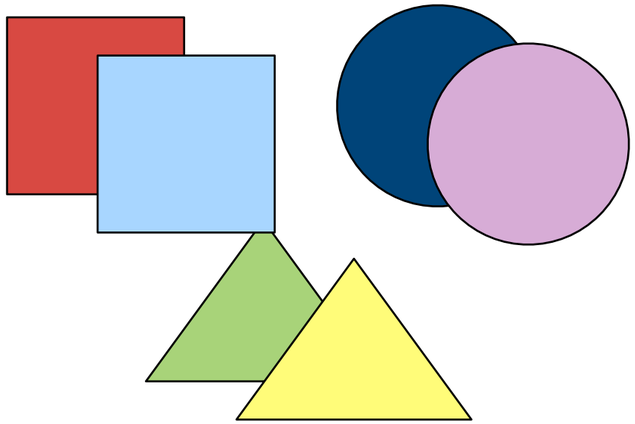
```
struct ShapeType {
    long   x;
    long   y;
    long   shapesize;
    string color;
};
#pragma keylist ShapeType color
```

Topic/Instances/Samples Recap.

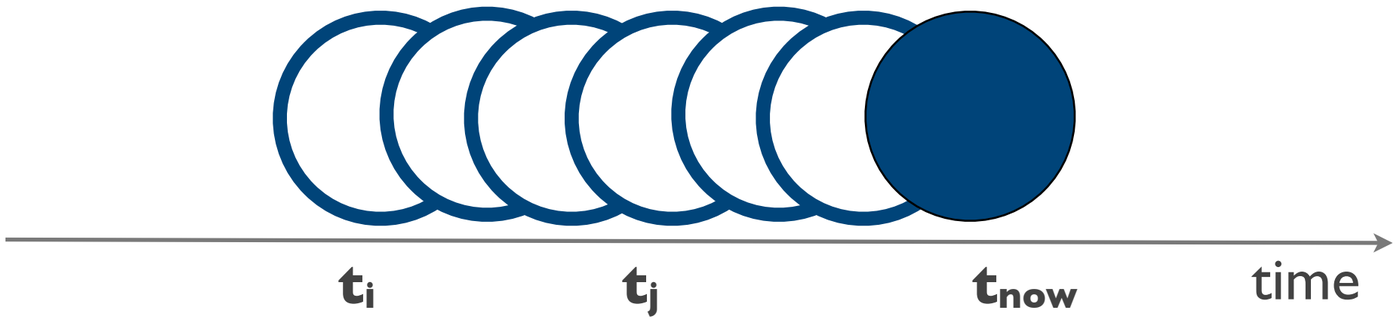
Topics



Instances



Samples



OpenSplice | DDS

Delivering Performance, Openness, and Freedom

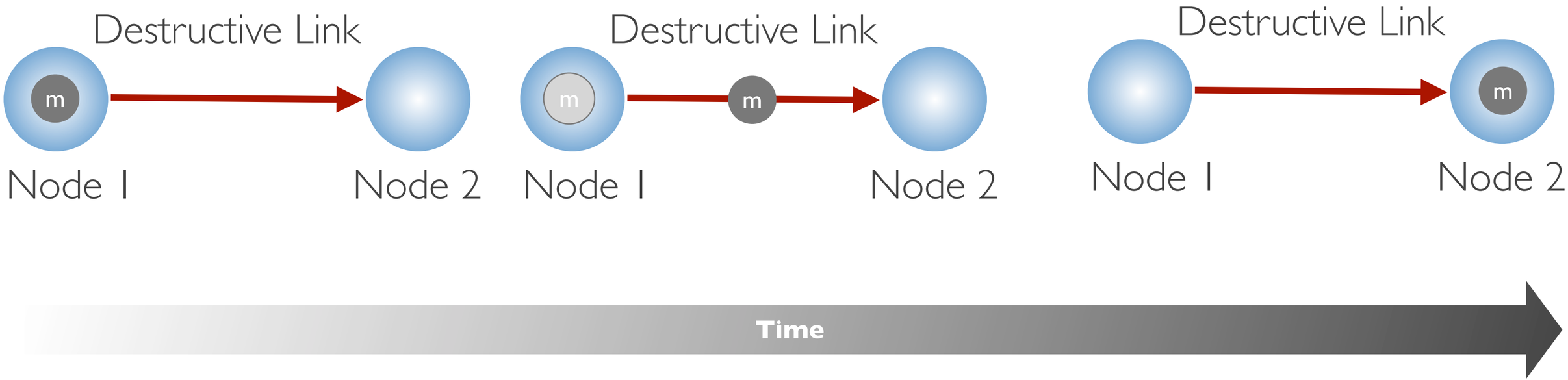
Node/Links vs. Reader/Writers

AMQP v1-0 Conceptual Model

- An AMQP Network consists of Nodes and Links
- A Node is a named source and/or sink of Messages. A Message is created at a (Producer) Node, and may travel along links, via other nodes until it reaches a terminating (Consumer) Node.
- A Link is a unidirectional route between nodes along which messages may travel. Links may have entry criteria (Filters) which restrict which messages may travel along them. The link lifetime is tied to the lifetime of the source and destination nodes.

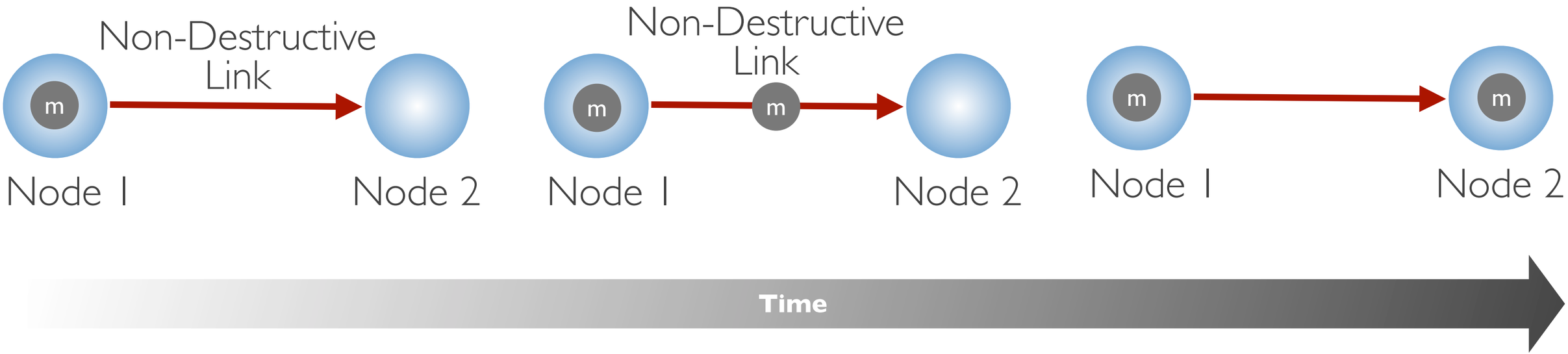
Destructive Links

Destructive Links consume the message from the originating source.



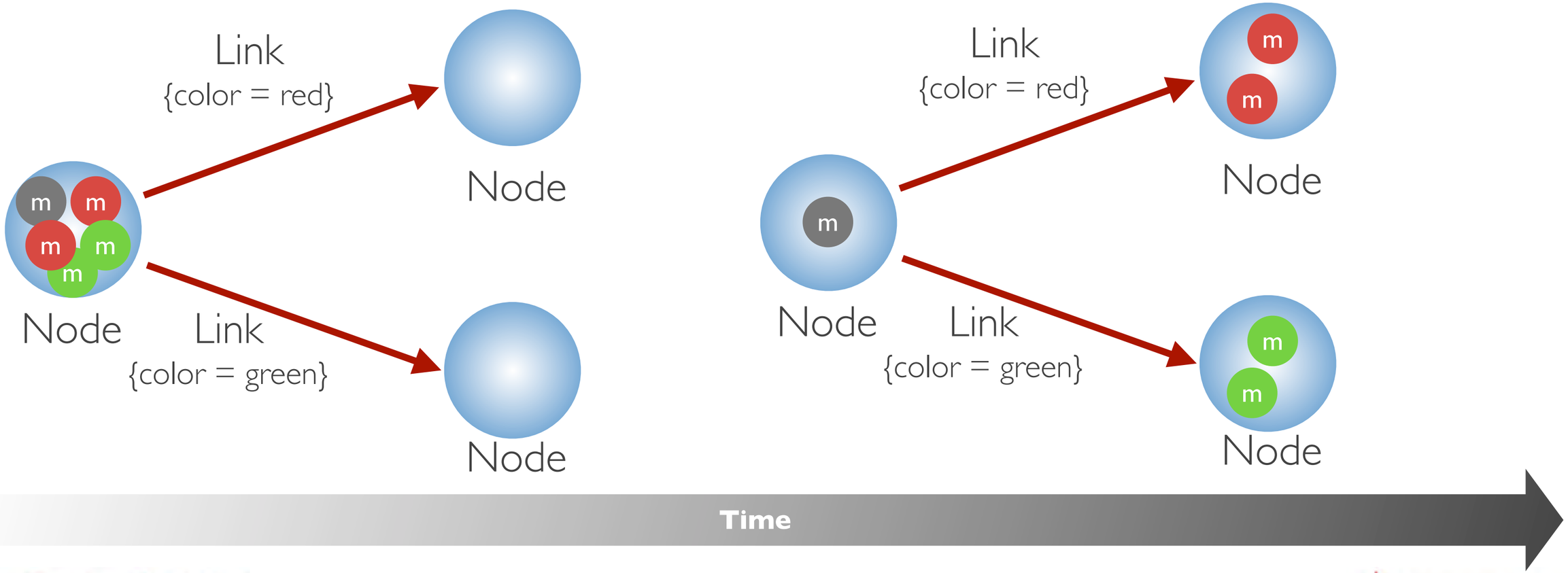
Non-Destructive Links

Non-Destructive Links don't consume messages from the source node.

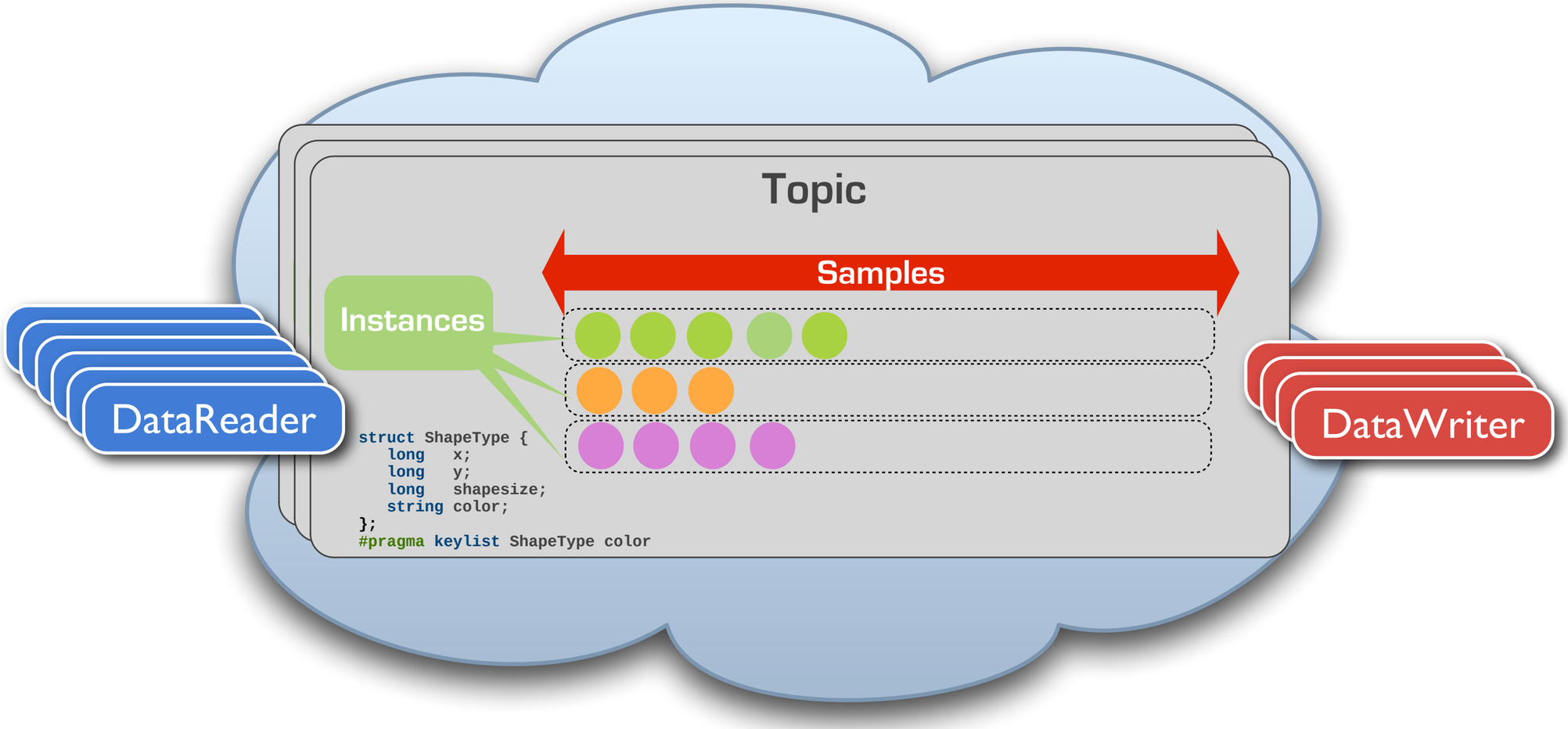


Filtered Links

Links can have associated filters that allow to predicate on the messages that might traverse. Filter can be based on the non opaque portion of the message.



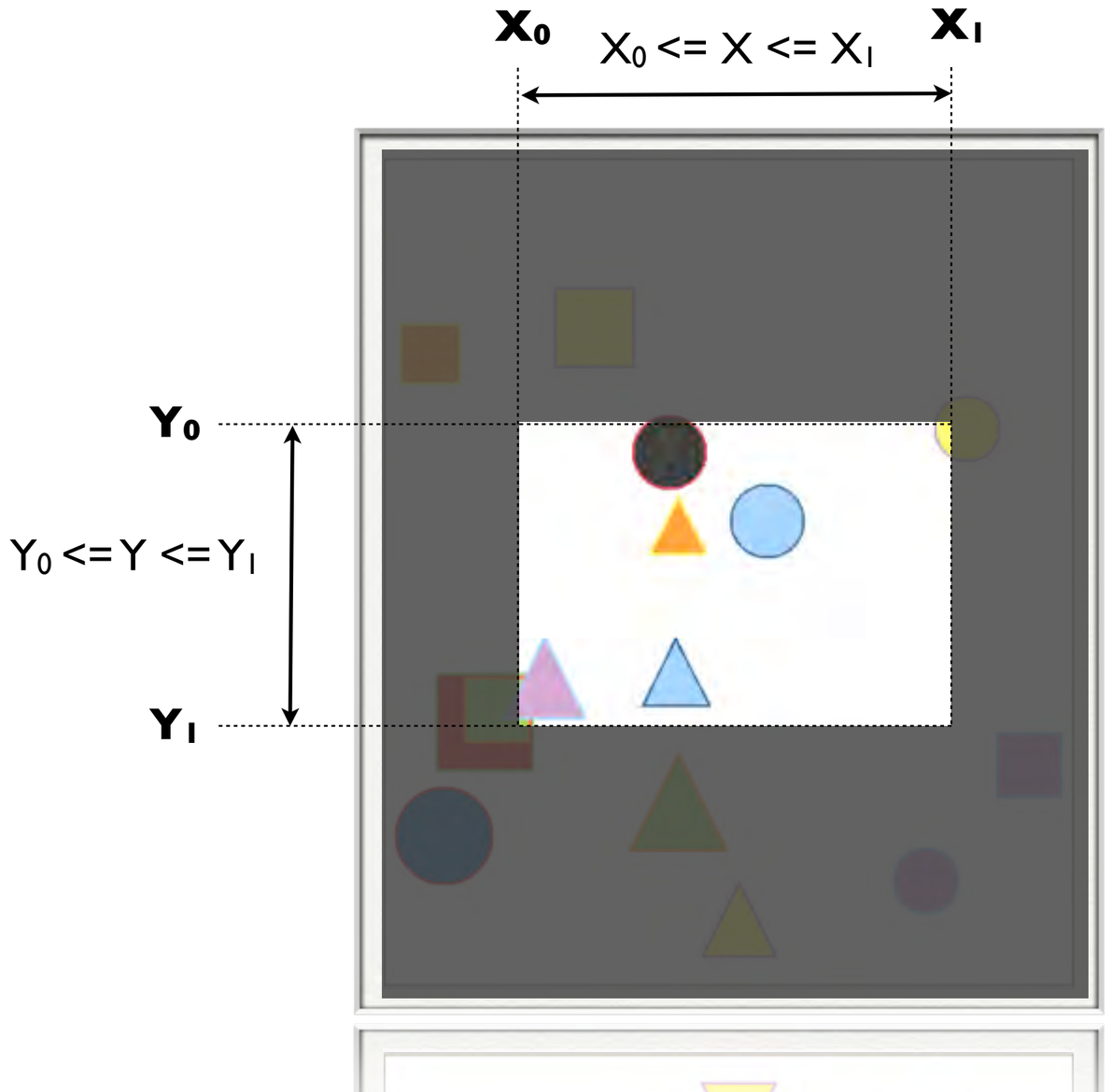
Anatomy of a DDS Application



Destructive/Non-Destructive Links can be paralleled with the DDS read/take semantics

Content Filtering

- DDS allows to specify **content-filtered Topics** for which a subset of SQL92 is used to express the filter condition
- Content filters can be applied on the entire content of the Topic Type
- Content filters are applied by DDS each time a new sample is produced/delivered



Local Queries

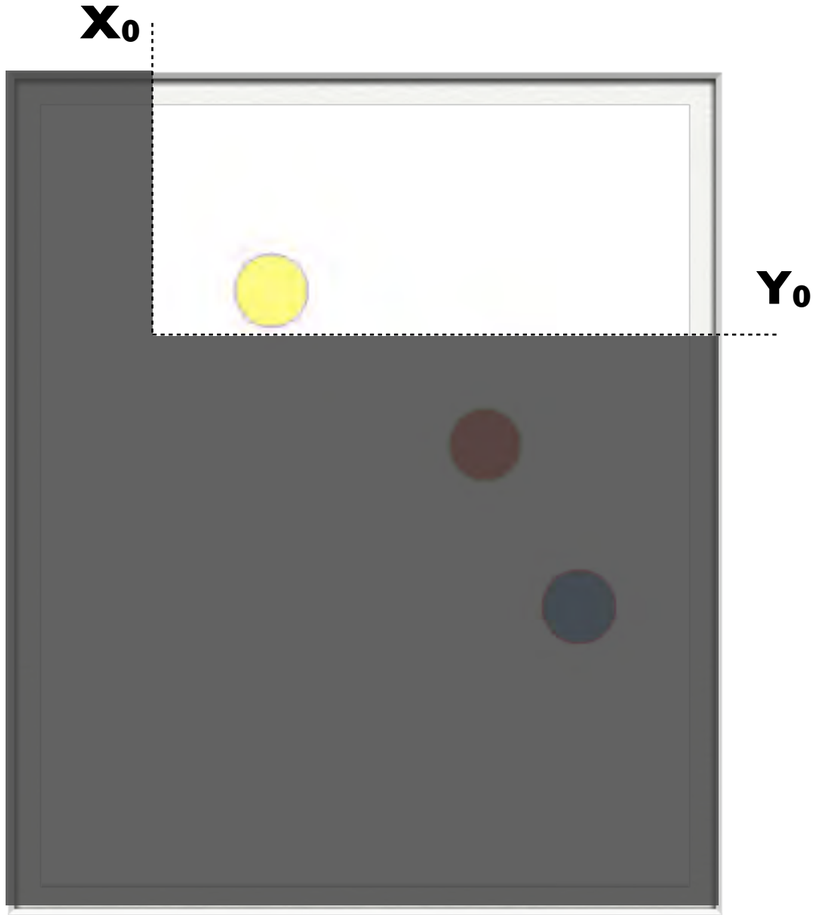
- A subset of SQL92 can be used for performing queries
- Queries are performed under user control and provide a result that depends on the current snapshot of the system, e.g., samples currently available

Circle Topic

color	x	y	shapesize
red	57	62	50
blue	90	85	50
yellow	30	25	50

x > 25 AND y < 55

color	x	y	shapesize
yellow	30	25	50



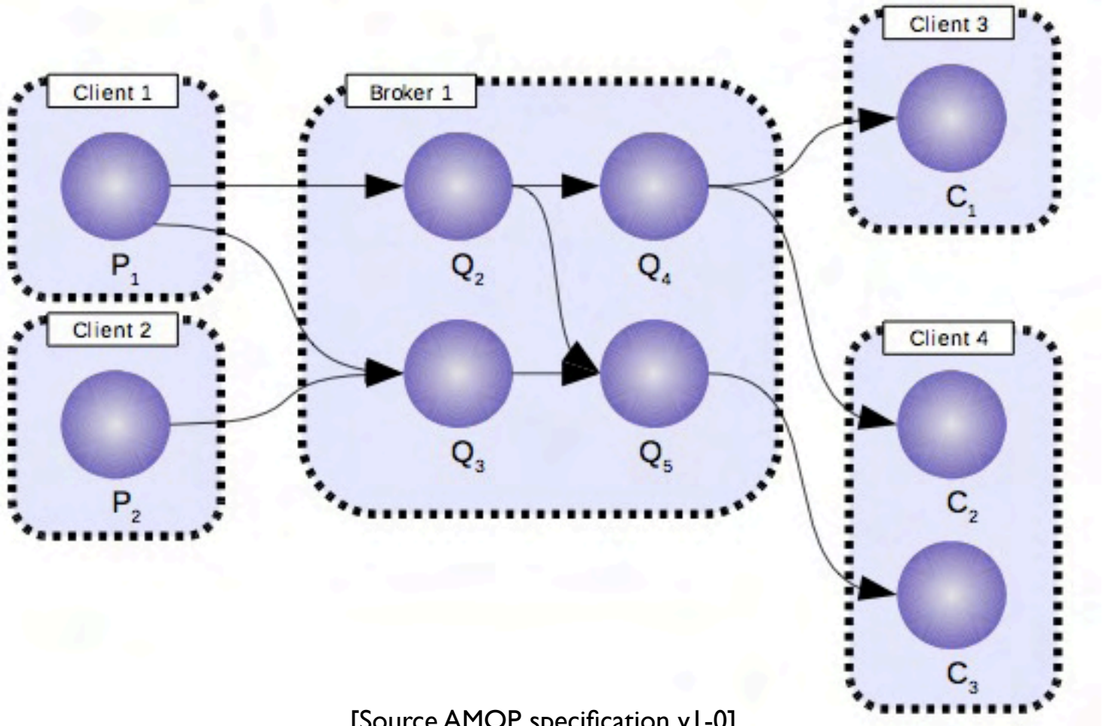
OpenSplice | DDS

Delivering Performance, Openness, and Freedom

Deployment Model

AMQP v1-0 Deployment Model

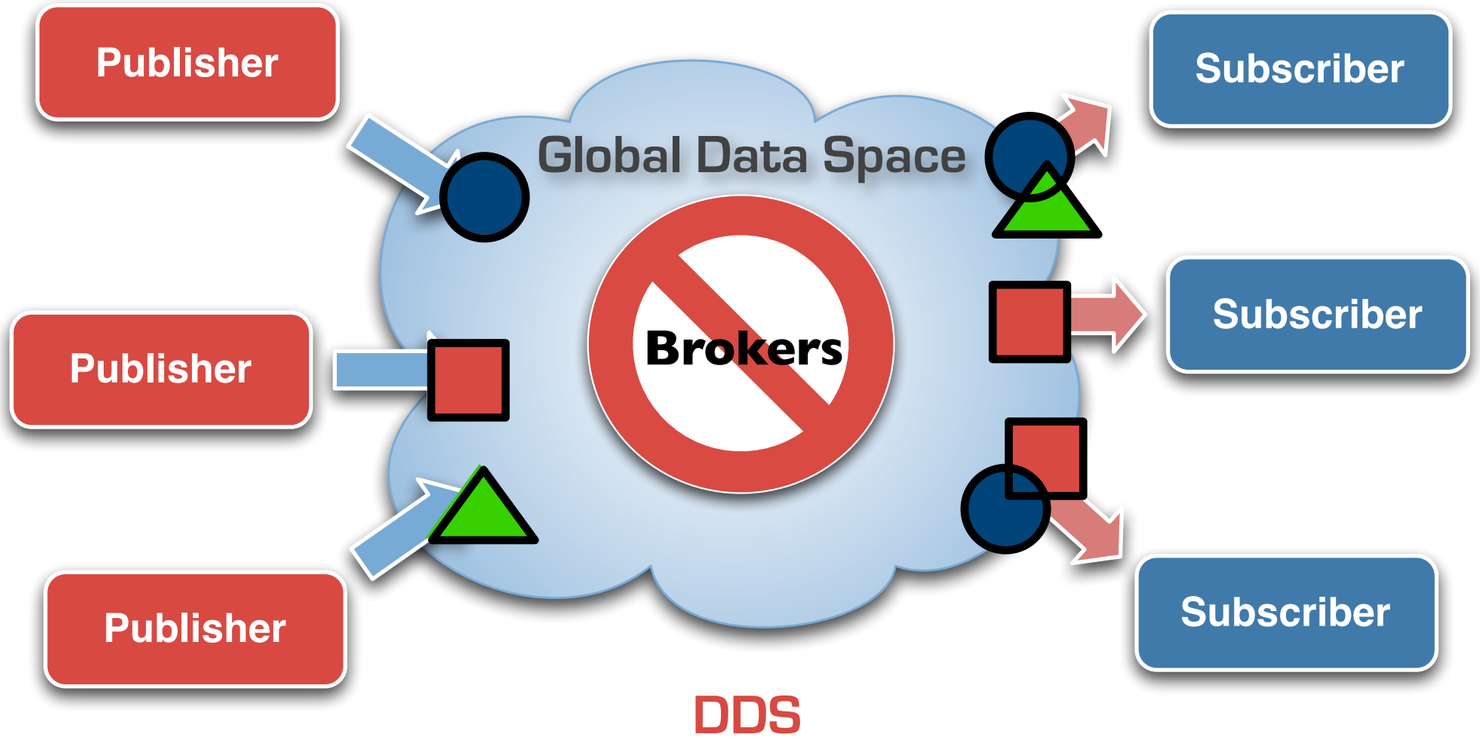
- Nodes within the AMQP network exist within Containers
- A container is a physical or logical process to which network connections can be established



[Source AMQP specification v1-0]

DDS Deployment Model

- DDS is based around the concept of a **fully distributed Global Data Space (GDS)**
- **Publishers** and **Subscribers** can join and leave the GDS at any time
- **Publishers and Subscribers** express their intent to **produce/consume specific type of data**, e.g., **Topics**
- **Data flows from Publisher to Subscribers**



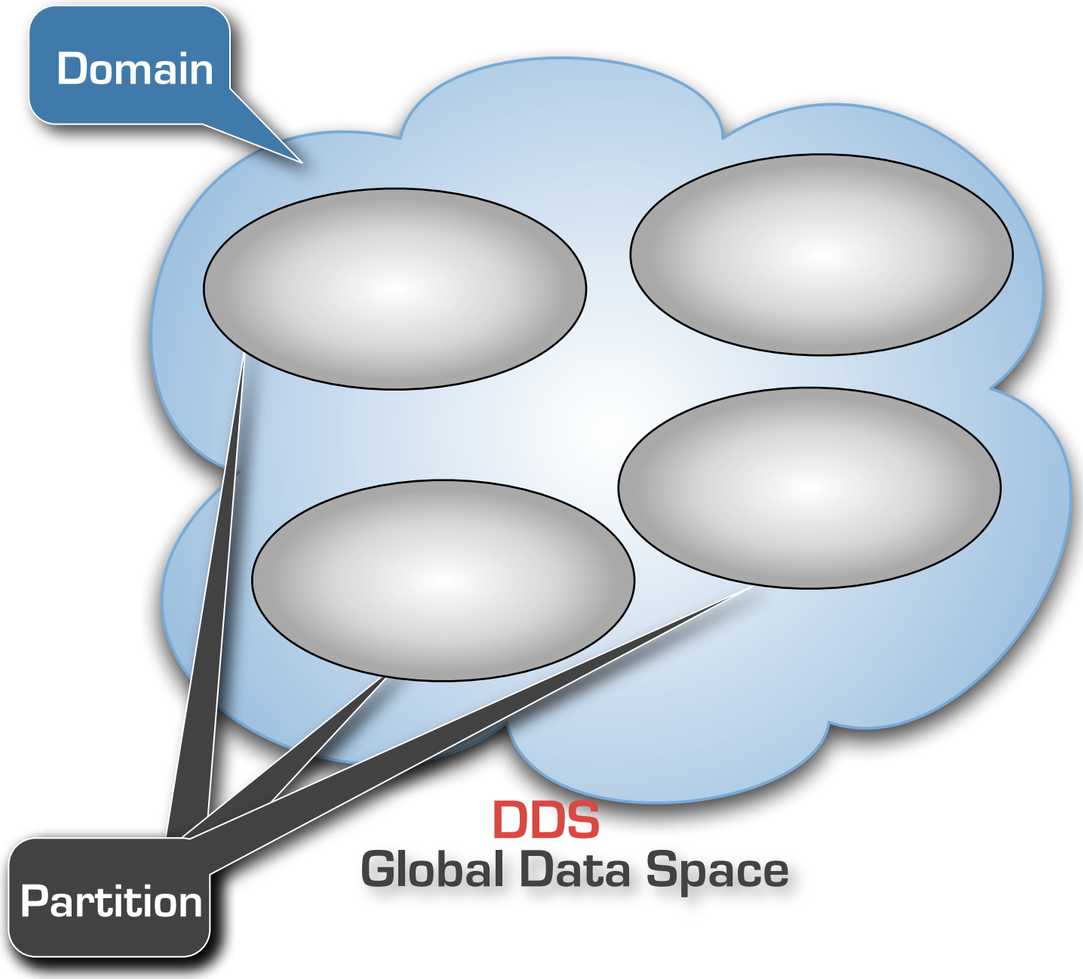
Domains and Partitions

Domain

- A Domain is one instance of the DDS Global Data Space
- DDS entities always belong to a specific domain

Partition

- A partition is a scoping mechanism provided by DDS organize a domain



OpenSplice | DDS

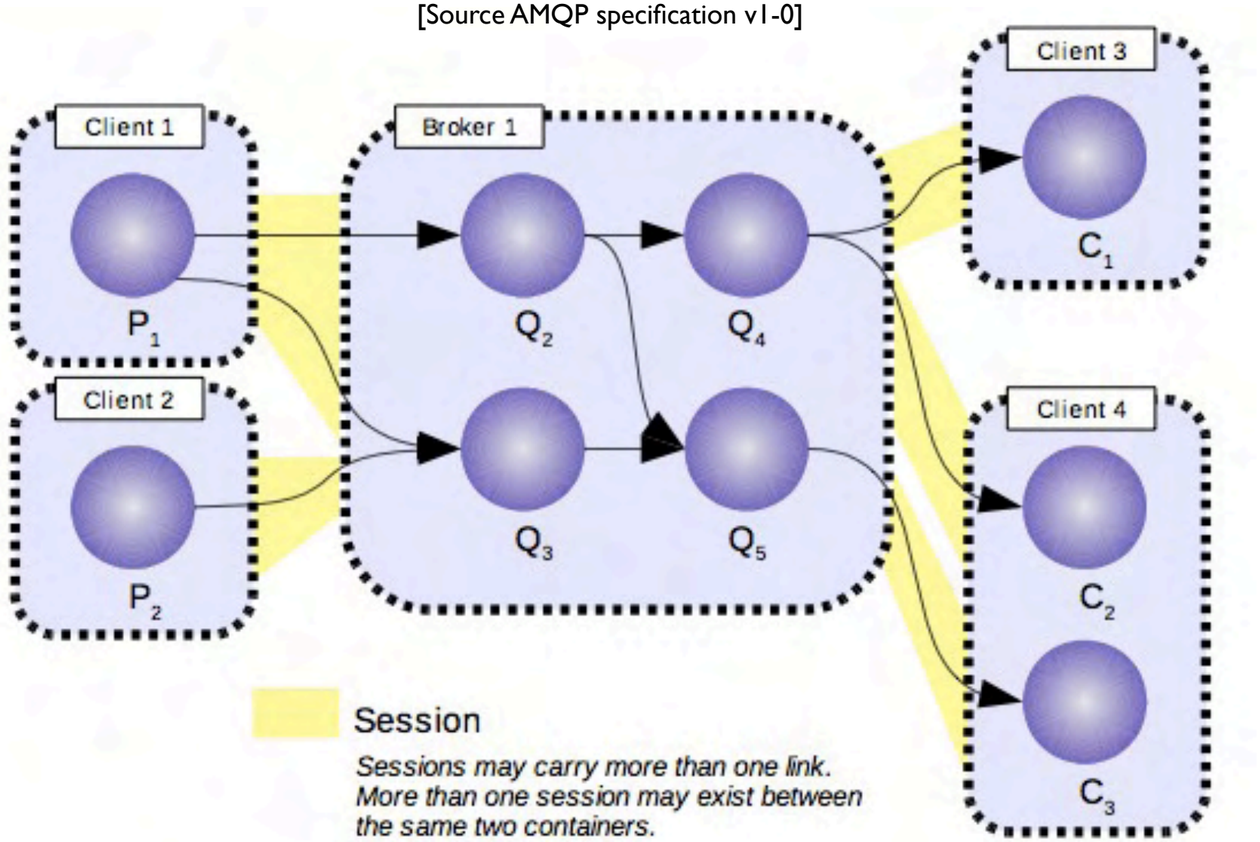
Delivering Performance, Openness, and Freedom

Communication Model

Sessions

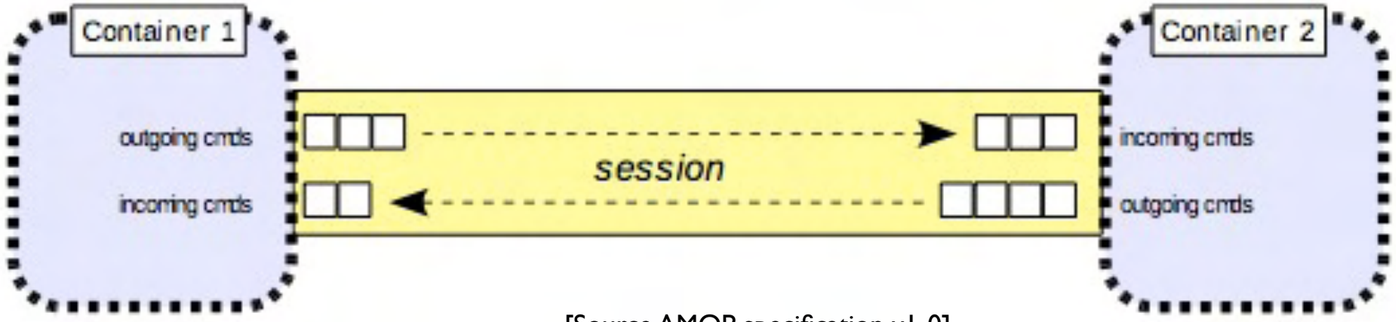
- A Session is a named interaction between two containers providing for a pair of reliable ordered command streams (one in each direction)
- Links between nodes in different containers are created on a session
- Containers and Sessions form an underlay network, nodes and links an overlay network atop them

[Source AMQP specification v1-0]



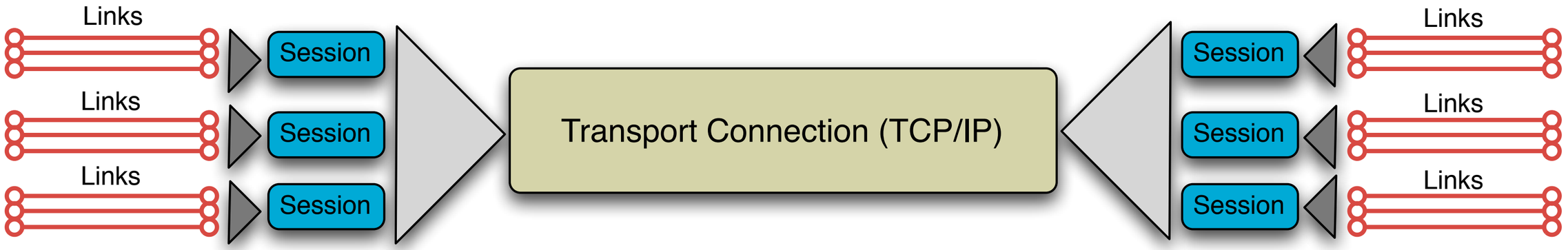
Sessions & Commands

- Sessions are a transport for commands
- Commands are the atomic units of work of the AMQP transport protocol Commands are used to create links between nodes in the source and destination containers, to transfer message data, and to issue and revoke credit
- In general an AMQP session will be carried over some form of network layer, thus commands sent on a session are asynchronous.

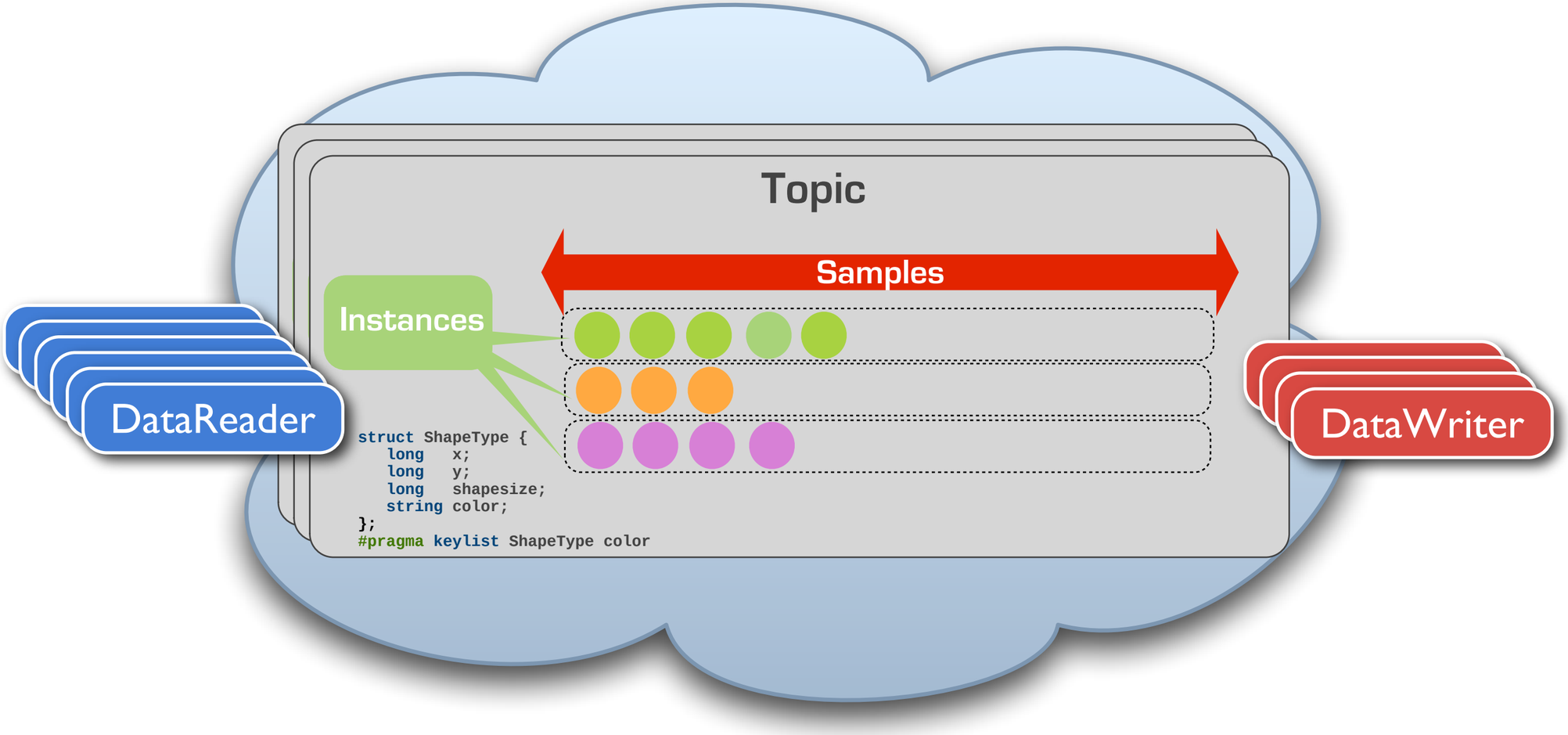


[Source AMQP specification v1-0]

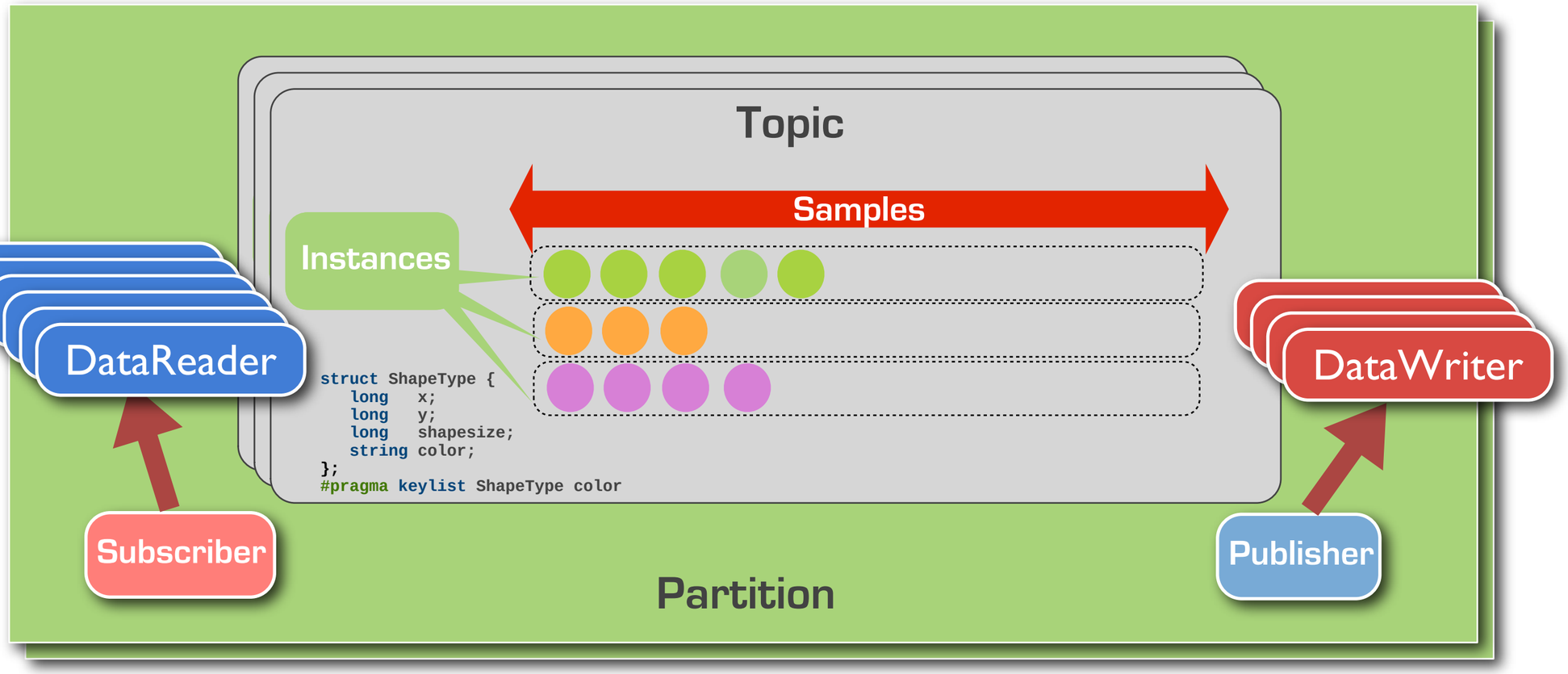
Transport Model



Anatomy of a DDS Application

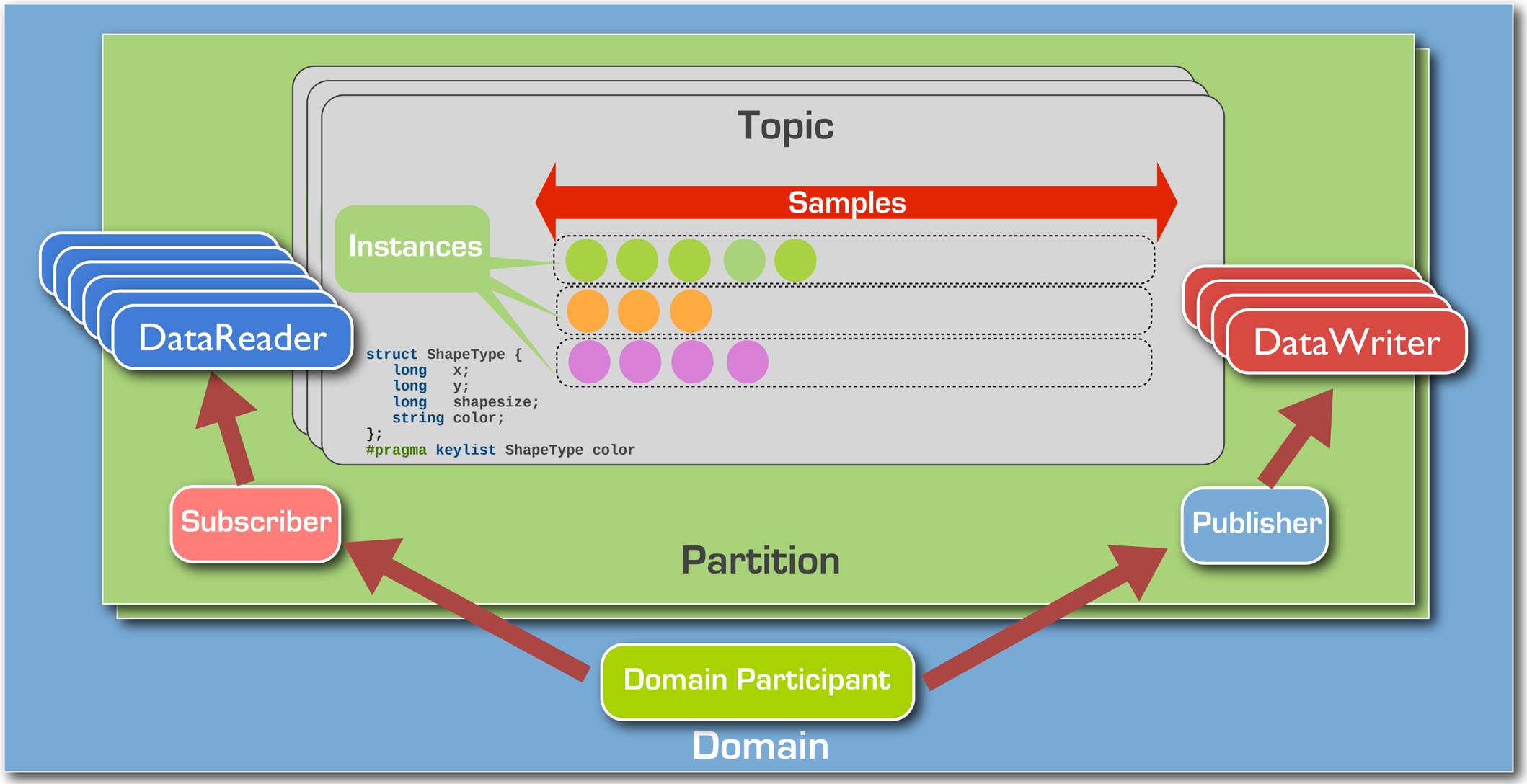


Anatomy of a DDS Application



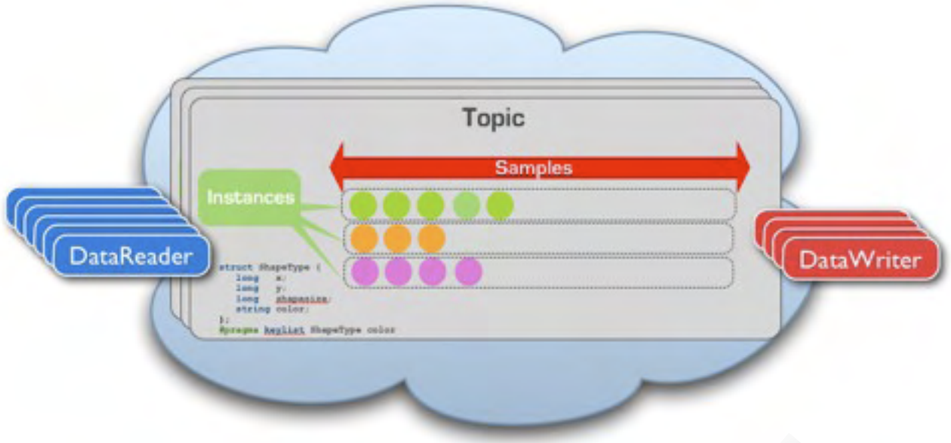
Arrows show structural relationships, not data-flows

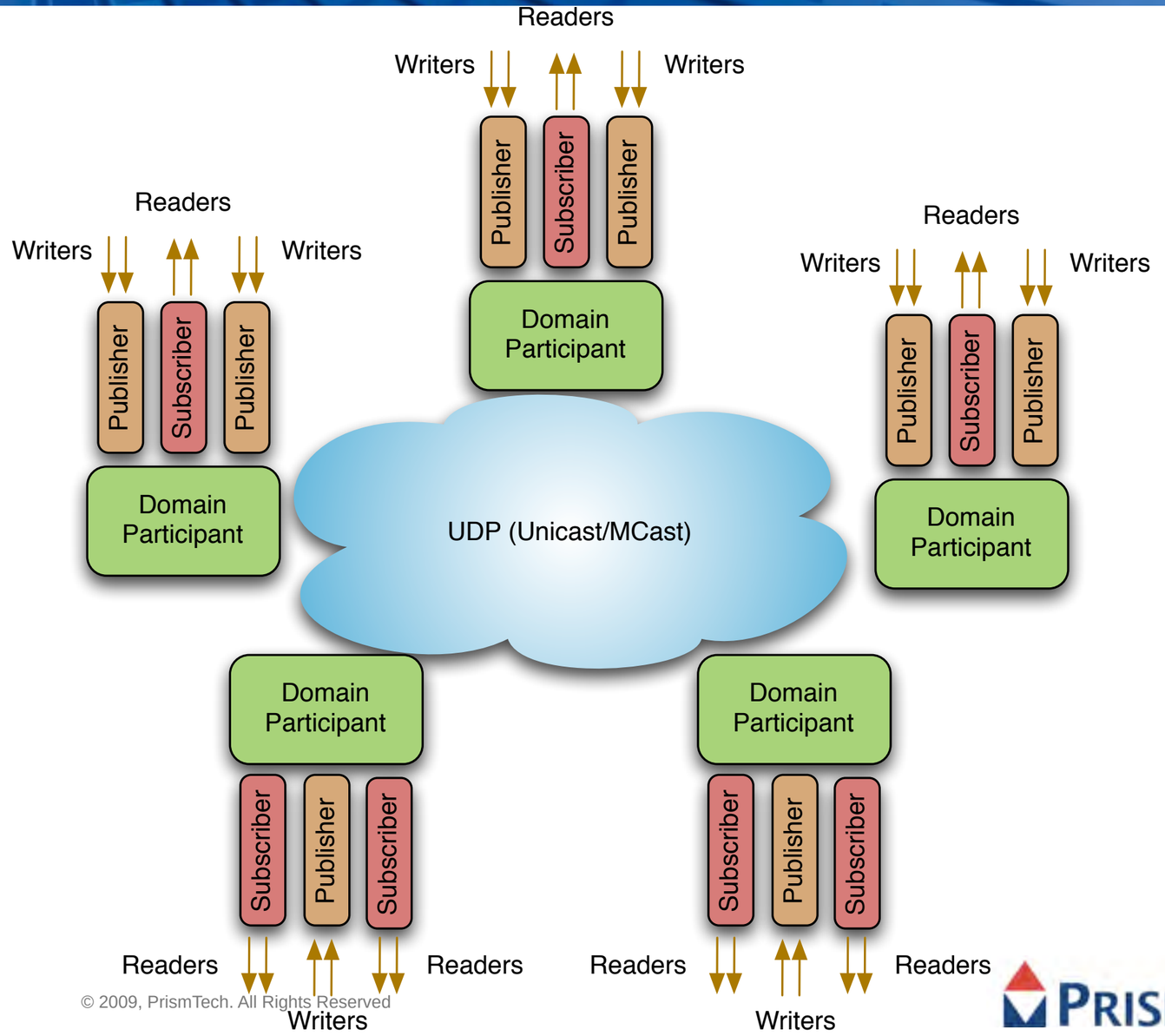
Anatomy of a DDS Application



Arrows show structural relationships, not data-flows

Transport Model





OpenSplice | DDS

Delivering Performance, Openness, and Freedom

Type System

AMQP Type System

- AMQP defines a type-system that is used to encode messages and controls sent as part of the protocol
- This type system can be also used to encode application data

null:	Indicates an empty value.
boolean:	Represents a true or false value.
ubyte:	Integer in the range 0 to 2^8 - 1.
ushort:	Integer in the range 0 to 2^16 - 1.
uint:	Integer in the range 0 to 2^32 - 1.
ulong:	Integer in the range 0 to 2^64 - 1.
byte:	Integer in the range -(2^7) to 2^7 - 1.
short:	Integer in the range -(2^15) to 2^15 - 1.
int:	Integer in the range -(2^31) to 2^31 - 1.
long:	Integer in the range -(2^63) to 2^63 - 1.
float:	32-bit floating point number (IEEE 754-2008 binary32).
double:	64-bit floating point number (IEEE 754-2008 binary64).
char:	A single unicode character.
timestamp:	An absolute point in time.
uuid:	A universally unique id as defined by RFC-4122 section 4.1.2.
binary:	A sequence of octets.
string:	A sequence of unicode characters.
symbol:	Symbols are values from a constrained domain. Although the set of possible domains is open-ended, typically the both number and size of symbols in use for any given application will be small, e.g. small enough that it is reasonable to cache all the distinct values.
list:	A sequence of polymorphic values.
map:	A polymorphic mapping from distinct keys to values.

[Source AMQP specification v1-0]

DDS Type System

- DDS supports the definition of types based on a subset of the IDL specification language.
- This type system is used internally by DDS and is also available to user for defining topic types

Primitive Types	
boolean	long
octet	unsigned long
char	long long
wchar	unsigned long long
short	float
unsigned short	double
	long double

Template Type	Example
string<length = UNBOUNDED>	string s1; string<32> s2;
wstring<length = UNBOUNDED>	wstring ws1; wstring<64> ws2;
sequence<T,length = UNBOUNDED>	sequence<octet> oseq; sequence<octet, 1024> oseq1k; sequence<MyType> mtseq; sequence<MyType, 10> mtseq10;
fixed<digits,scale>	fixed<5,2> fp; //d1d2d3.d4d5

Constructed Types	Example
enum	enum Dimension { 1D, 2D, 3D, 4D };
struct	struct Coord1D { long x;}; struct Coord2D { long x; long y; }; struct Coord3D { long x; long y; long z; }; struct Coord4D { long x; long y; long z, unsigned long long t;};

union	union Coord switch (Dimension) {
	case 1D:
	Coord1D c1d;
	case 2D:
	Coord2D c2d;
	case 3D:
	Coord3D c3d;
	case 4D:
	Coord4D c4d;
	};

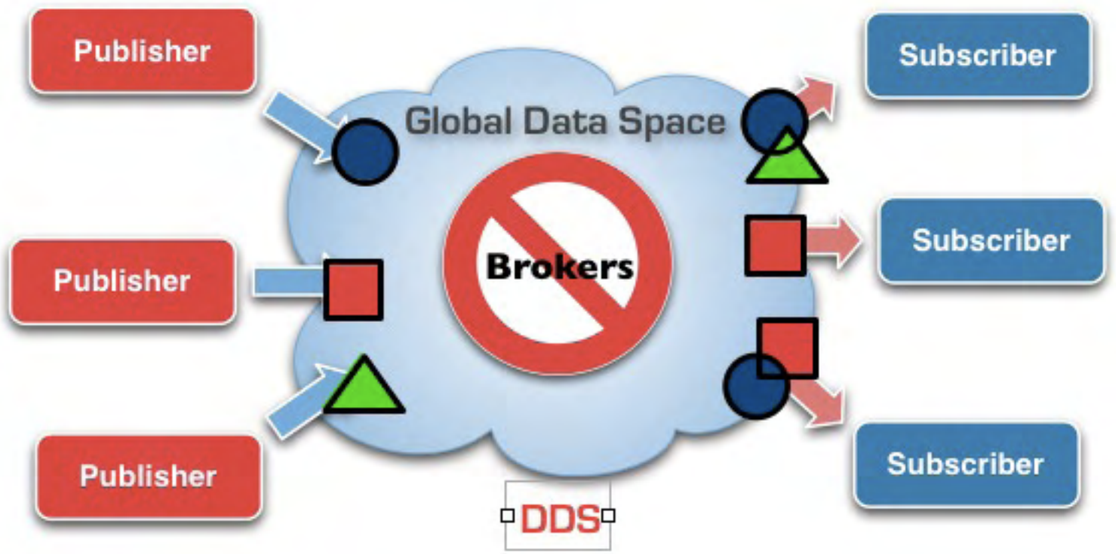
OpenSplice | DDS

Delivering Performance, Openness, and Freedom

Putting it all Together

DDS

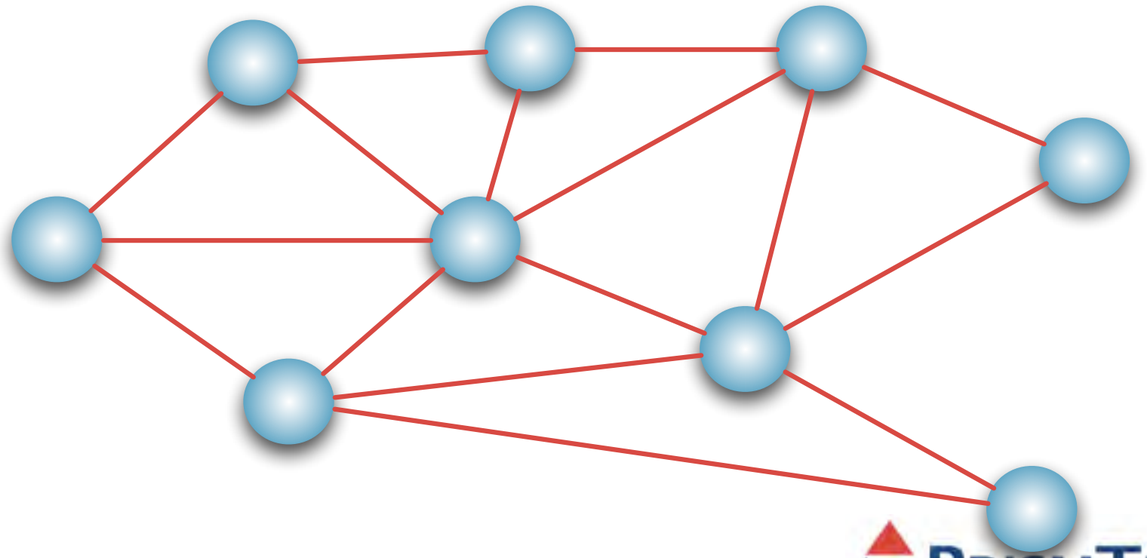
- A generic DDS application is a self-formed federation reads and writes topics over the Global Data Space
- Communication is hidden to the applications which is provided with a read/write sematics
- UDP Unicast and Multicast are available for DDS v1.2, DDSI v2.1



OpenSplice DDS

AMQP

- A generic AMQP application is a graph of nodes connected by links over which travel application provided messages
- The link-related traffic is managed by sessions which in turns communicate via a transport connection.
- The only transport currently available for AMQP v1-0 is TCP



PRISMTECH

DDS

- **Data Centricity.** DDS is about data and it is fair to say that data takes life in DDS, thanks to the support for keys, lifecycle management, etc.
- **Transport Protocol.** Currently supports UDP (Unicast and Multicast). TCP will be supported in upcoming revisions of the standard. However, there is nothing that prevents DDS today to use TCP as a transport
- **Topology Configuration.** Topology is dynamically discovered by DDS via its standard discovery protocol

AMQP

- **Addressing Scheme.** Does not define a standard global addressing scheme, this will be part of v1-1
- **Transport Protocol.** Currently only supports TCP, but UDP and SCTP will be added in future revisions of the spec
- **Broker Management.** A SIG is working on defining an API for Broker Management, today, each implementation has its own way
- **Topology Configuration.** The Broker topology can be configured via management tools, yet the spec does not explicitly supports automatic propagation of queue subscriptions (some product do)

OpenSplice | DDS

Delivering Performance, Openness, and Freedom

Code Examples

Setting-up Subscriptions

DDS

```
dds::Topic<Track> track("TrackTopic");
dds::Topic<TrackClass> track_class("TrackClassTopic");

dds::DataReader<Track> tDR(track);
dds::DataReader<TrackClass> tcDR(track_class);
```

```
struct Track {
   short id;
   long x;
   long y;
};
#pragma keylist Track id
```

```
struct TrackClass {
   short id;
   Classification cls;
   string descr;
};
#pragma keylist TrackClass id
```

AMQP

```
std::string exchange = "amq.topic";

// Connect to the broker
Connection connection;
connection.open(host, port);

// Create a session
Session session =  connection.newSession();

// Decrlare a queue
session.queueDeclare(arg::queue=queue,
                     arg::exclusive=true,
                     arg::autoDelete=true);

// Set-up routes
session.exchangeBind(arg::exchange=exchange,
                     arg::queue="Track",
                     arg::bindingKey="Track.#");
session.exchangeBind(arg::exchange=exchange,
                     arg::queue="TrackClass",
                     arg::bindingKey="TrackClass.#");
```

Writing Data

DDS

```
dds::DataWriter<Track> tDW(track);
dds::DataWriter<TrackClass> tcDW(track_class);

Track t = { 101, 140, 200 };
tDW(t);

TrackClass tc = { 101, FRIENDLY, “Some descr.”};
tcDW.write(tc);
```

AMQP

```
Track t = { 101, 140, 200 };
Message message;

message.getDeliveryProperties().setRoutingKey("Track.Radar101");

std::stringstream message_data;
data << t.id << “ “ << t.x << “ “ << t.y;

message.setData(message_data.str());
async(session).messageTransfer(arg::content=message,
                               arg::destination="amq.topic");
```

Reading Data

DDS

```
std::vector<Track> data;
std::vector<SampleInfo> info;

tDR.read(data, info);
```

AMQP

```
void Listener::received(Message& message) {
   std::string data = message.getData();
   Track t;
   // Parse the string to extract the data
   string_to_track(data, t);
}
```

OpenSplice | DDS

Delivering Performance, Openness, and Freedom

Summing Up

Concluding Remarks

DDS

- DDS provides the abstraction of a Global Data Space, a ubiquitous, universal and fully distributed data cache.
- DDS makes user data a first class citizen
- DDS provides a standard API as well as an interoperable Wire-Protocol

AMQP

- AMQP is provides as basic abstraction that of messages and message routing.
- AMQP standardized the wire-protocol and uses this standard to achieve API independence

DDS and AMQP are very interesting technologies, which at some point might work in synergy.

OpenSplice DDS could be using AMQP as one of its wire-protocols. The DDS API over AMQP could be standardize...

Online Resources

OpenSplice DDS
Delivering Performance, Openness, and Freedom

* http://www.opensplice.com/
* emailto:opensplicedds@prismtech.com

slideshare
Present Yourself

* http://www.slideshare.net/angelo.corsaro

webex™

* http://bit.ly/1Sreg

twitter

* http://twitter.com/acorsaro/

YouTube

* http://www.youtube.com/OpenSpliceTube

Blogger

* http://opensplice.blogspot.com