A Deep Dive into the DDS/OPC UA Gateway Specification

Speaker:

Dr. Gerardo Pardo-Castellote,
Chief Technology Officer,
RTI

Moderator:

Brandon Lewis, OpenSystems Media

Agenda

- Housekeeping
- Presentation
- Questions and Answers
- Wrap-up

Your systems.
Working as one.

A Deep Dive into the DDS/OPC UA Gateway Specification

Gerardo Pardo-Castellote, Ph.D.
Chief Technology Officer, RTI

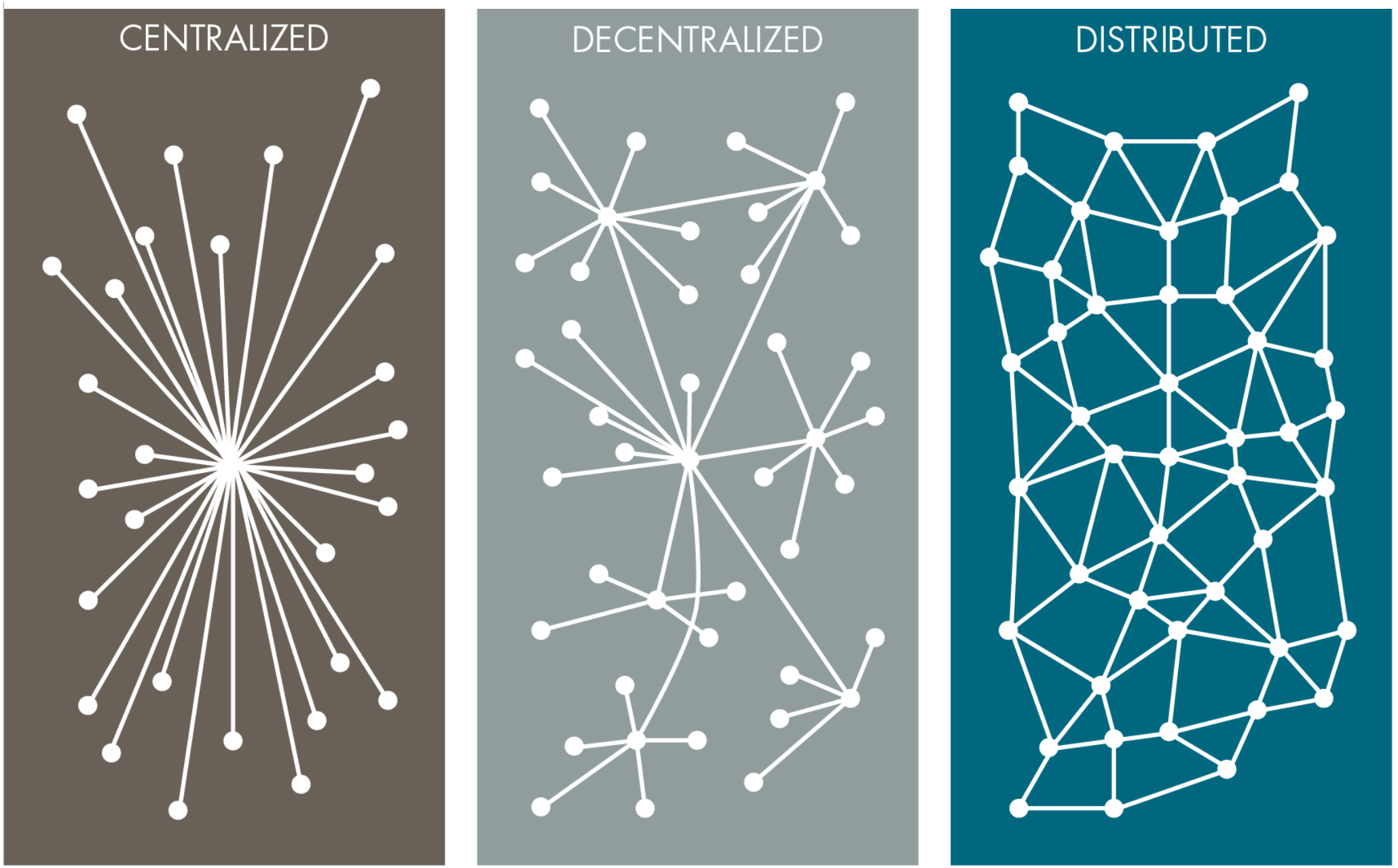
Fernando Garcia-Aranda
Senior Software Engineer, RTI

May 2018

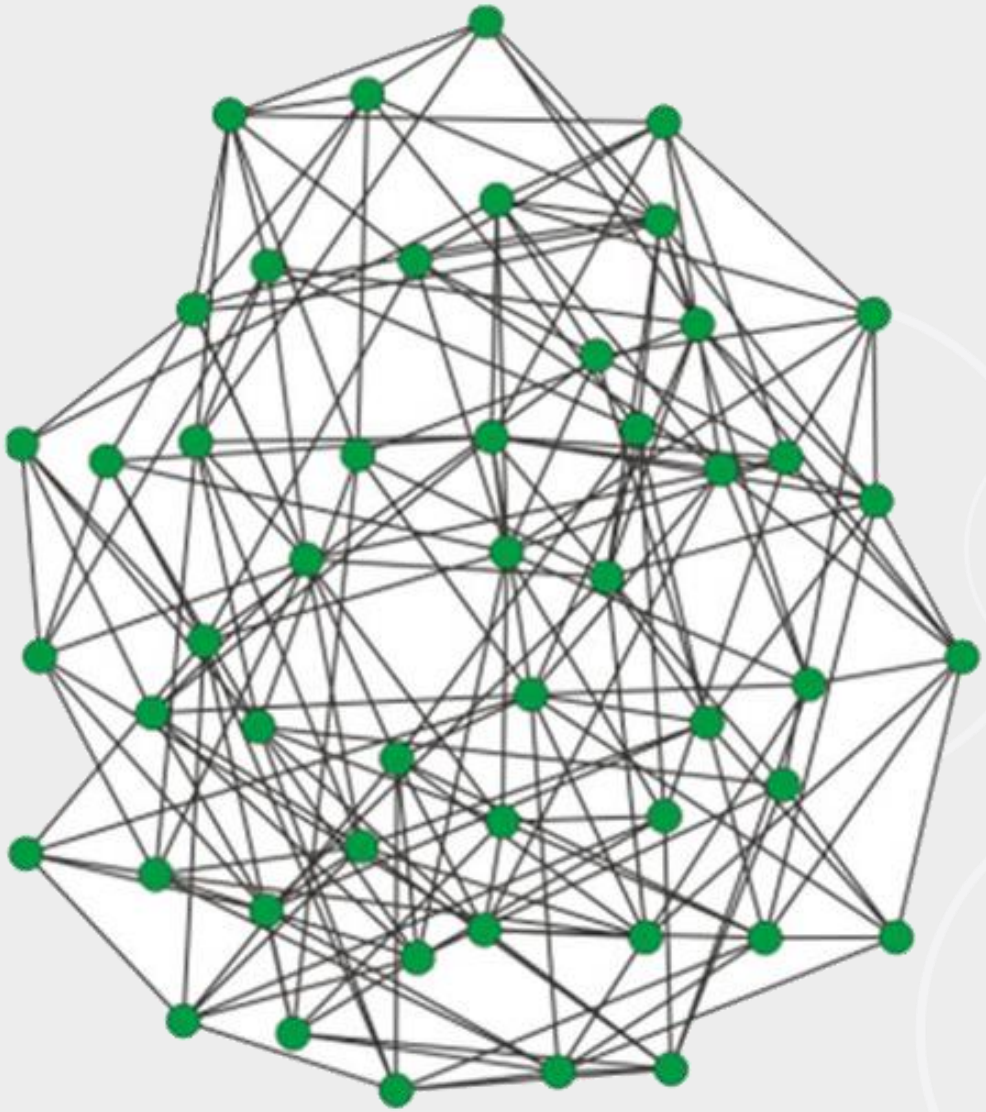
Outline

- Motivation
- Use-Cases
- OPC UA Concepts
- DDS Concepts
- OPC UA / DDS Gateway
- Conclusions

DECENTRALIZED PEER TO PEER SYSTEMS



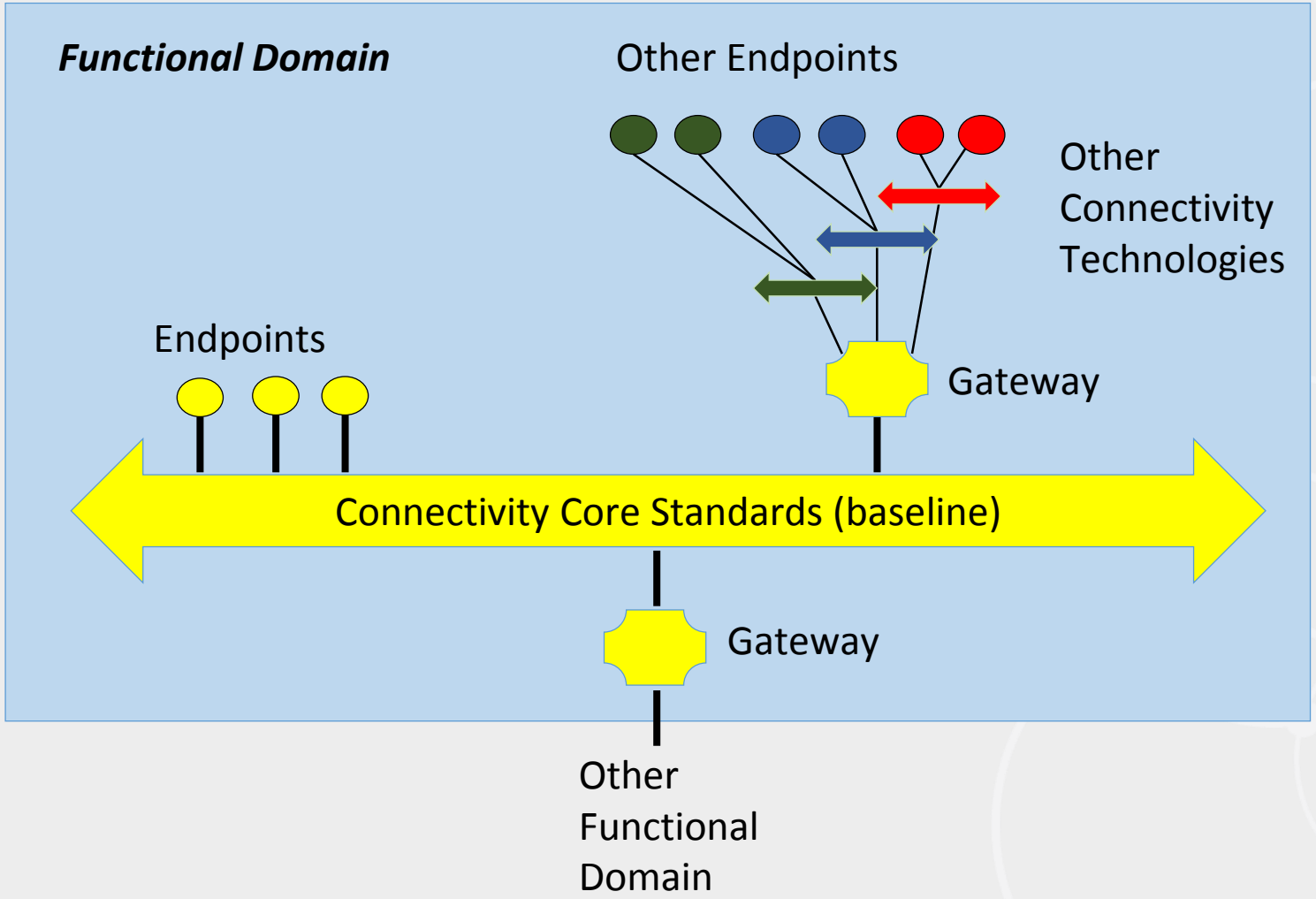
How to connect and integrate these systems?

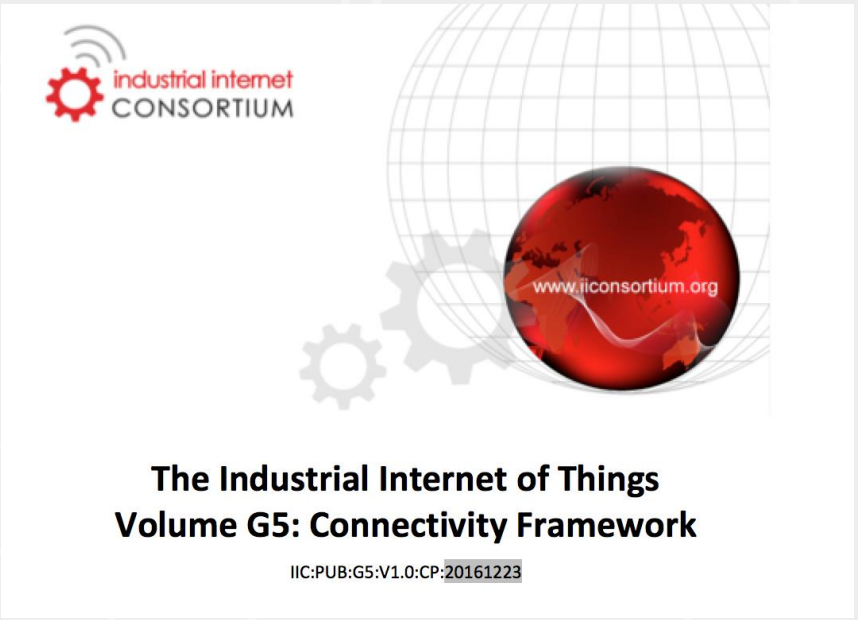


- Performance
- Scalability
- Reliability
- Redundancy
- Fail over
- Security
- Heterogeneity
- Domain-Specific Technologies
- Legacy Systems

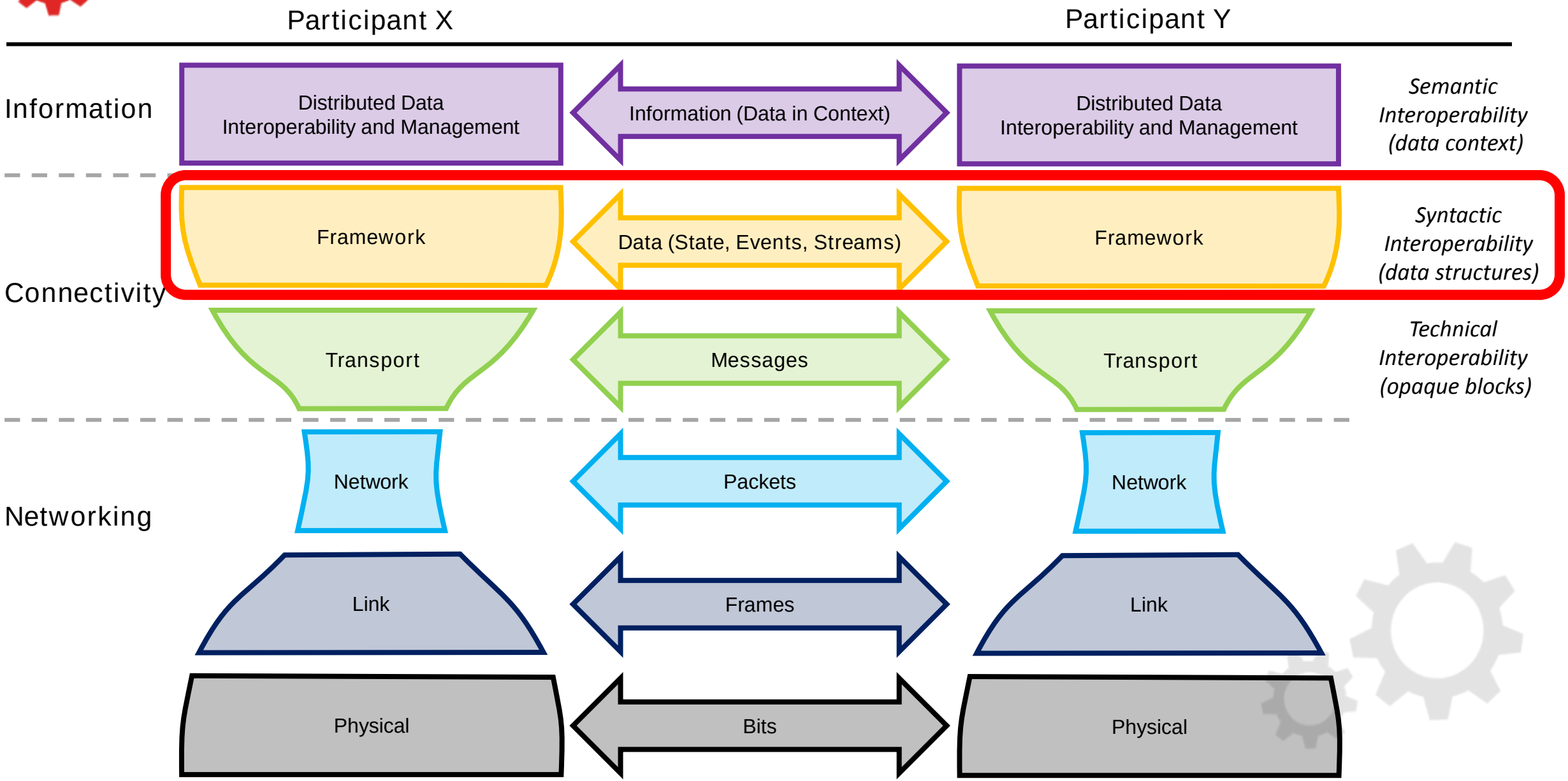
Industrial Internet Consortium: 270+ Companies, 30+ Countries

IIC Layered Databus architecture

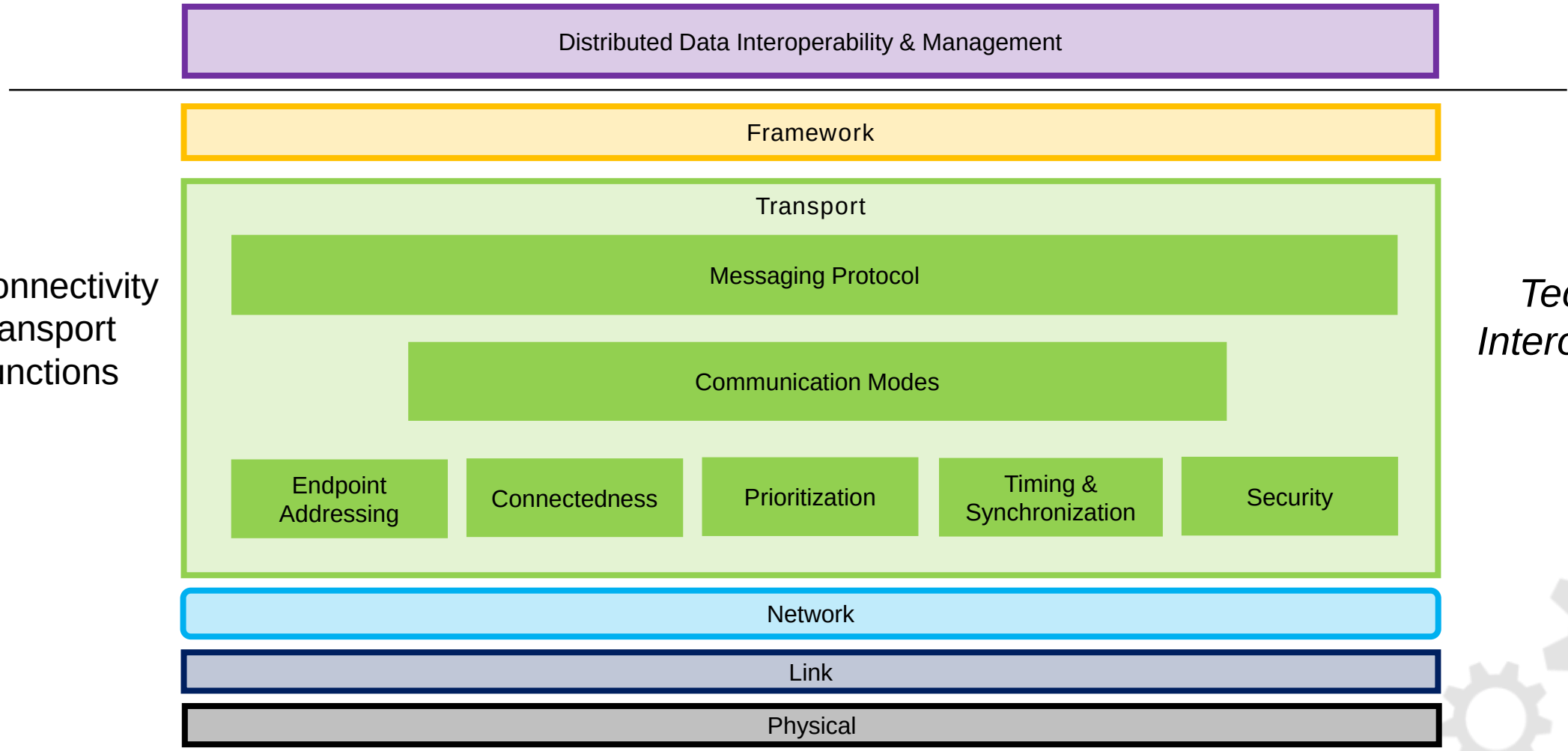




IIoT Connectivity Stack Model



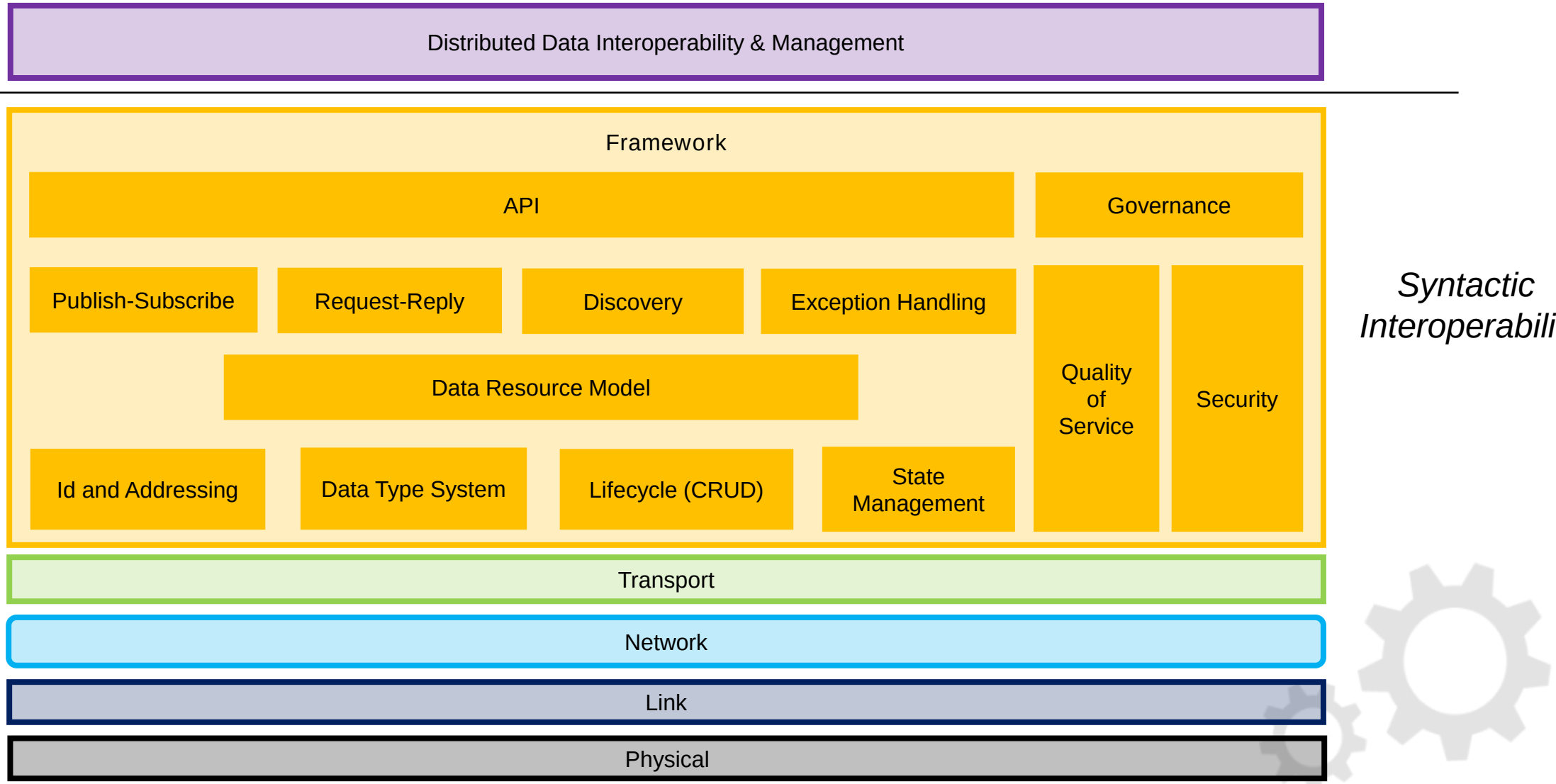
Connectivity Transport Layer



Connectivity Transport Functions

Technical Interoperability

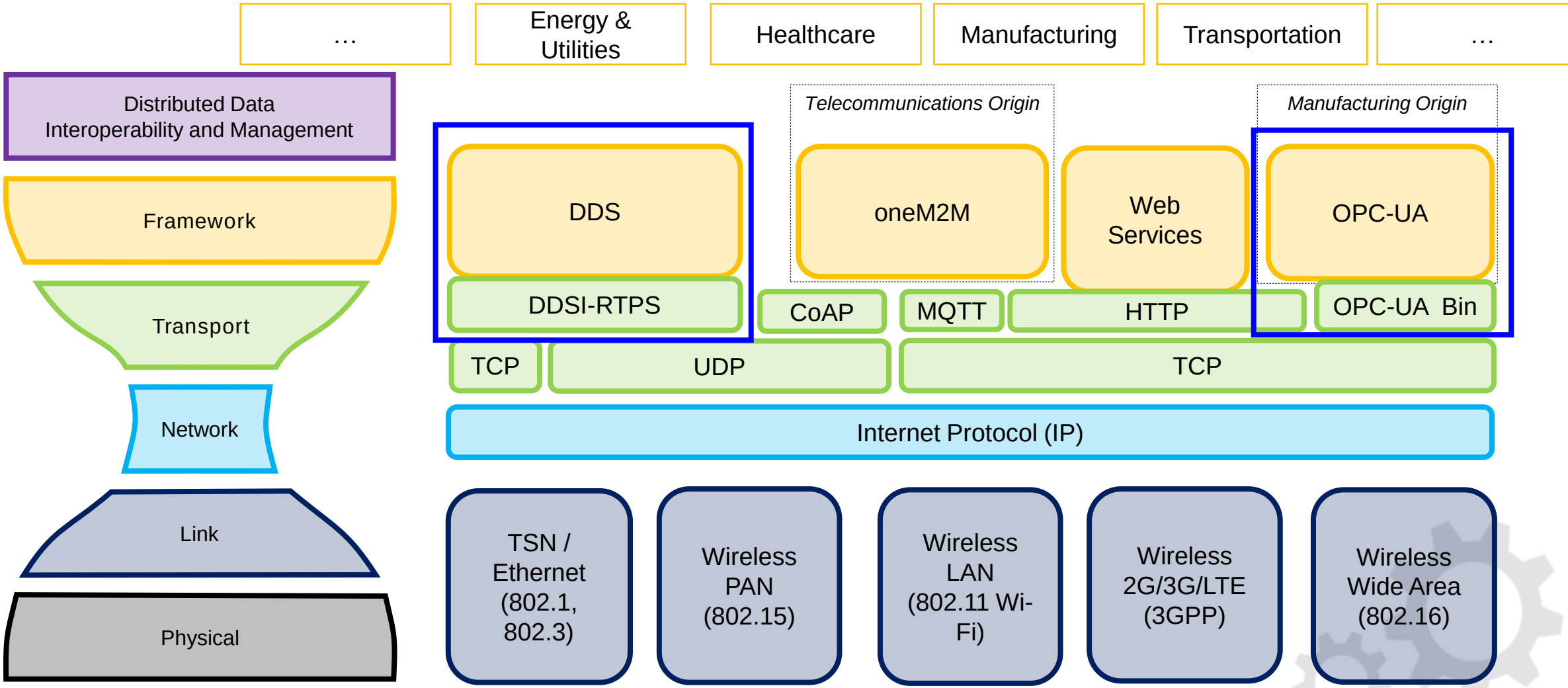
Connectivity Framework Layer



Connectivity Framework Functions

Syntactic Interoperability

Connectivity Standards



Selection Criteria

	Core Standard Criterion	DDS	Web Services	OPC-UA	oneM2M
1	Provide **syntactic interoperability**	✓	Need XML or JSON	✓	✓
2	Open standard with strong **independent, international** governance	✓	✓	✓	✓
3	**Horizontal** and neutral in its applicability across industries	✓	✓	✓	✓
4	**Stable** and **proven** across multiple vertical industries	Software Integration & Autonomy	✓	Manufacturing	Smart City Pilots*
5	Have **standards-defined *Core Gateways*** to *all* other core connectivity standards	Web Services, OPC-UA, oneM2M*	DDS, OPC-UA, oneM2M	Web Services, DDS, oneM2M*	Web Services, DDS*
6	Meet the connectivity framework **functional** requirements	✓	✗	Pub-Sub in development	
7	Meet **non-functional** requirements of performance, scalability, reliability, resilience	✓	✗	Real-time in development	Reports not yet documented or public
8	Meet **security** and safety requirements	✓	✓	✓	✓
9	Not require any single component from any single vendor	✓	✓	✓	✓
10	Have readily-available SDKs both **commercial** and **open source**	✓	✓	✓	✓

* = work in progress , ✓ = supported, ✗ = **not** supported

GREEN = Gating Criteria

DDS and the Industrial Internet of Things

Deployed in 1000s of Systems












Industries: Energy, Industrial Control, Transportation, Healthcare, Defense

Industrial IoT Systems

- Reliability: Severe consequences if offline for 5ms (or 5 min)
- Real-time: measure in ms or µs
- Interface scale: 10+ applications/teams
- Dataflow complexity: data has many destinations
- Architecture: Next generation IIoT

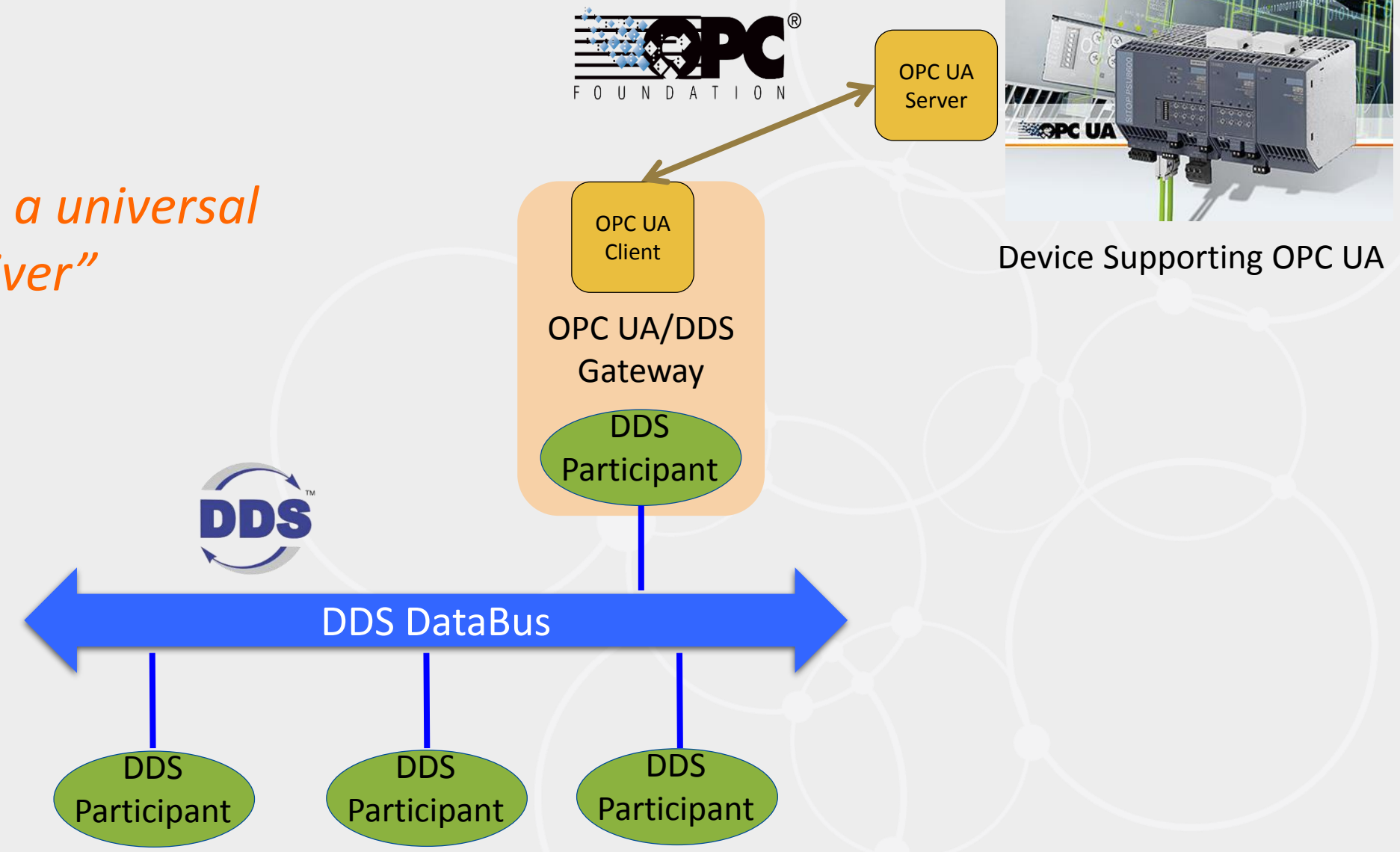
3+ Yes? → DDS

OSA · FACE Future Airborne Capability Environment · OPEN CONNECTIVITY FOUNDATION · THE OPEN GROUP OPEN PROCESS AUTOMATION FORUM · SGIP SMART GRID INTEROPERABILITY PANEL · ICE Alliance Integrating Clinical Environments · ROS · MD PnP Getting connected for patient safety · OpenFog ti

Use Cases

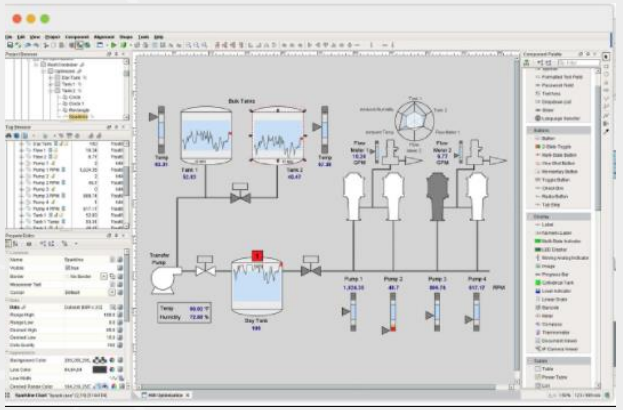
Scale access to OPC UA devices

OPC UA as a universal "device driver"

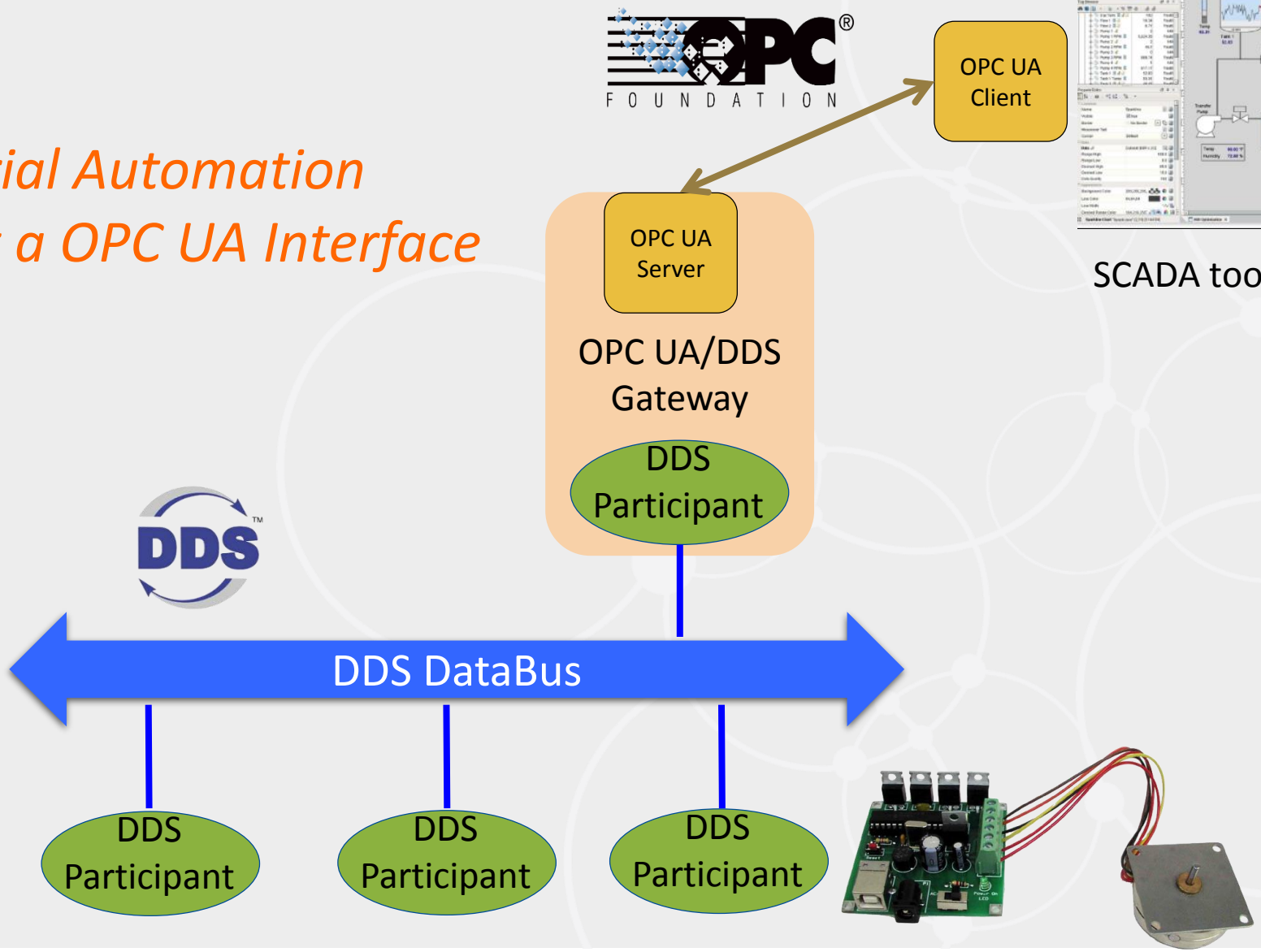


Use SCADA Tools in DDS systems

Many Industrial Automation Tools support a OPC UA Interface

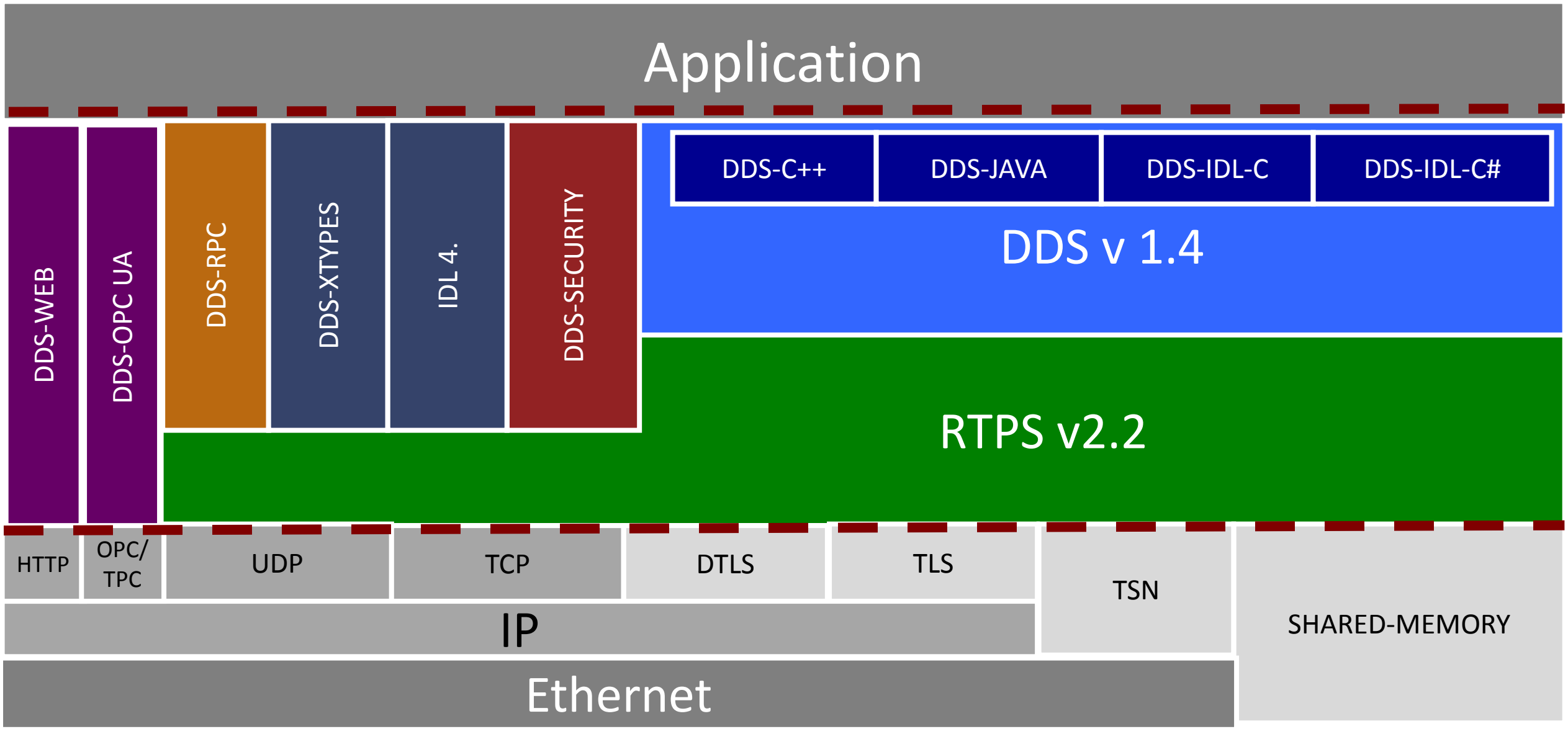


SCADA tool supporting OPC UA

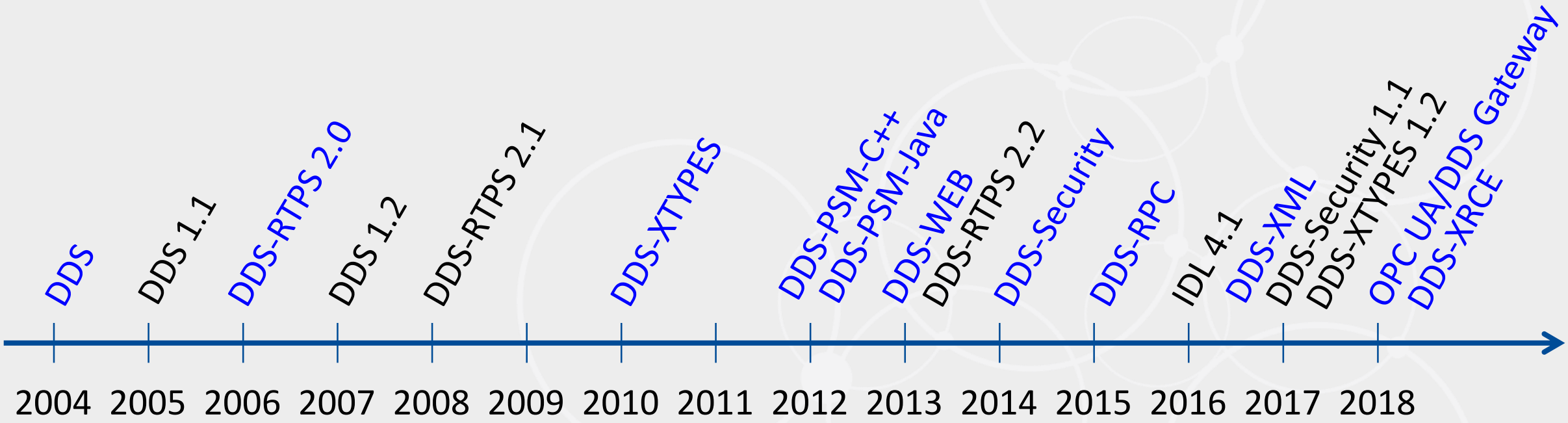


DDS Concepts

DDS Standard family

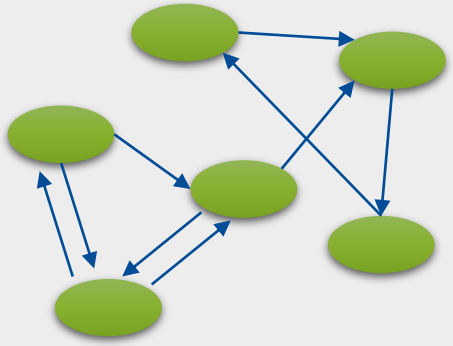


Timeline



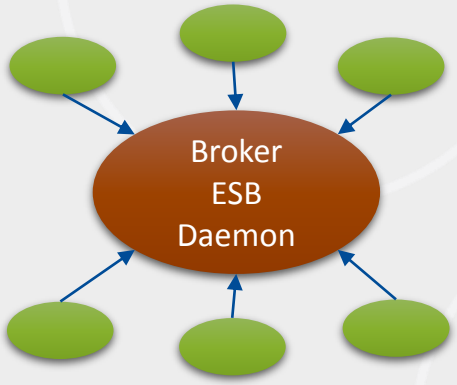
4th Gen Middleware: Data-Centric Publish-Subscribe

Point-to-Point
Client/Server



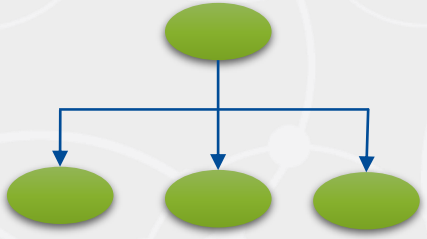
TCP, REST, WS*,
OPC UA

Brokered
Publish/Subscribe
Queuing



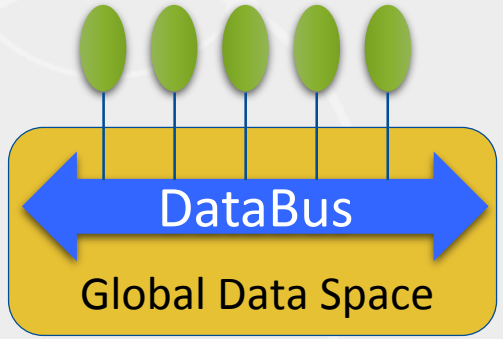
MQTT, XMPP
AMQP

Broadcast
Publish/Subscribe



Fieldbus, CANbus,
SOMEIP,
OPC UA Pub-Sub

Data-Centric
Publish-Subscribe



DDS (Data-Distribution
Service)

Virtual Global Data Space

DDS DOMAIN

Topic A

QoS

Topic B : "Turbine State"

QoS

Source (Key)	Speed	Power	Phase
WPT1	37.4	122.0	-12.20
WPT2	10.7	74.0	-12.23
WPTN	50.2	150.07	-11.98

Topic C

QoS

DDS

Topic D

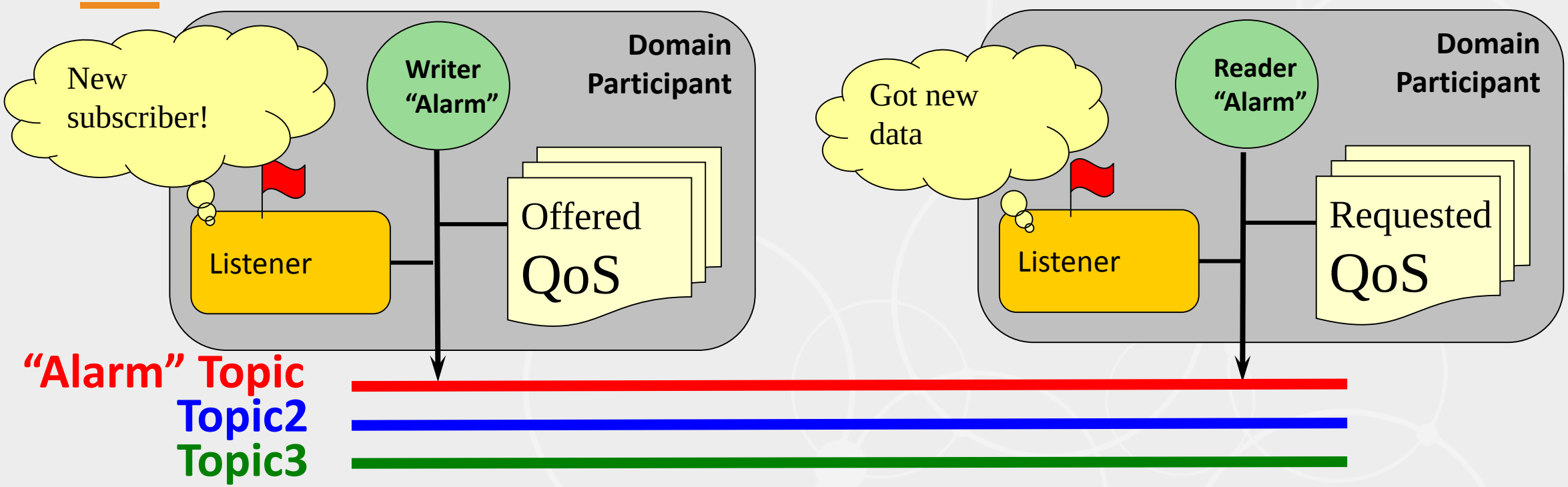
QoS

CRUD operations

Persistence Service

Recording Service

Data-Centric Communications Model



- **Participants** scope the global data space (domain)
- **Topics** define the data-objects (collections of subjects)
- **DataWriters** publish data on Topics
- **DataReaders** subscribe to data on Topics
- **QoS Policies** are used configure the system
- **Listeners** are used to notify the application of events

Request <= Offered QoS compatibility checking and run-time monitoring

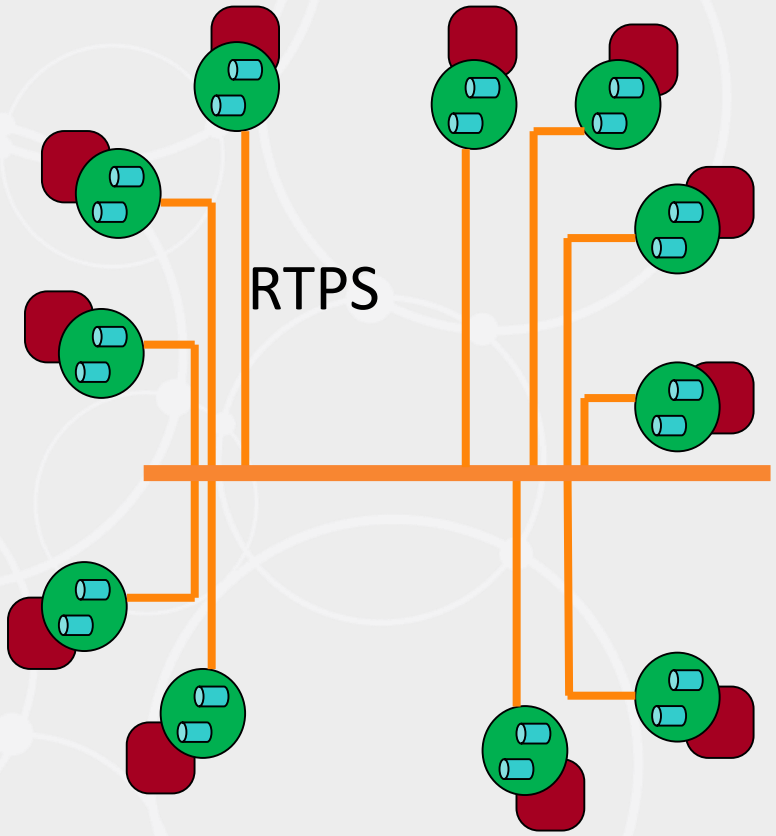
Quality of Service (QoS) Policies

	QoS Policy
Cache	**DURABILITY**
	HISTORY
	LIFESPAN
Resources	WRITER DATA LIFECYCLE
	READER DATA LIFECYCLE
	ENTITY FACTORY
	RESOURCE LIMITS
Delivery	**RELIABILITY**
	TIME BASED FILTER
	DEADLINE
	CONTENT FILTERS

QoS Policy	
USER DATA	User QoS
TOPIC DATA	
GROUP DATA	
PARTITION	Presentation
PRESENTATION	
DESTINATION ORDER	
OWNERSHIP	Availability
OWNERSHIP STRENGTH	
LIVELINESS	
LATENCY BUDGET	Transport
TRANSPORT PRIORITY	

RTPS Protocol optimized for IIOT

- Full peer-to-peer protocol
  - No required brokers or servers
- Adaptable via QoS
  - Reliability, timeouts, message priority
- Native reliable multicast support
  - Uses transport multicast, if available, else unicast UDP
- Robust to disconnects
  - Maintains session above (UDP) transport
- Efficient data encapsulation
  - Binary XCDR
- Built-in availability and durability
  - Durable & Persistent data, Historical cache, Failover support



Data and Service Definition

DDS-XTYPES and IDL4 standards

- Logical Data Model and Service Interfaces
  - **Portable**: Language-Independent Type System
  - **Safe**: Rules for Type Compatibility
  - **Flexible**: Types/Interfaces expressed in IDL or XML
- Interoperable **System Evolution**
  - Types/Services changes **(add, remove, reorder, …)**
  - Incremental/Partial upgrades
- **Dynamic** API's to access data and types
  - Systems that adapt at run-time
- **Efficient** binary serialization

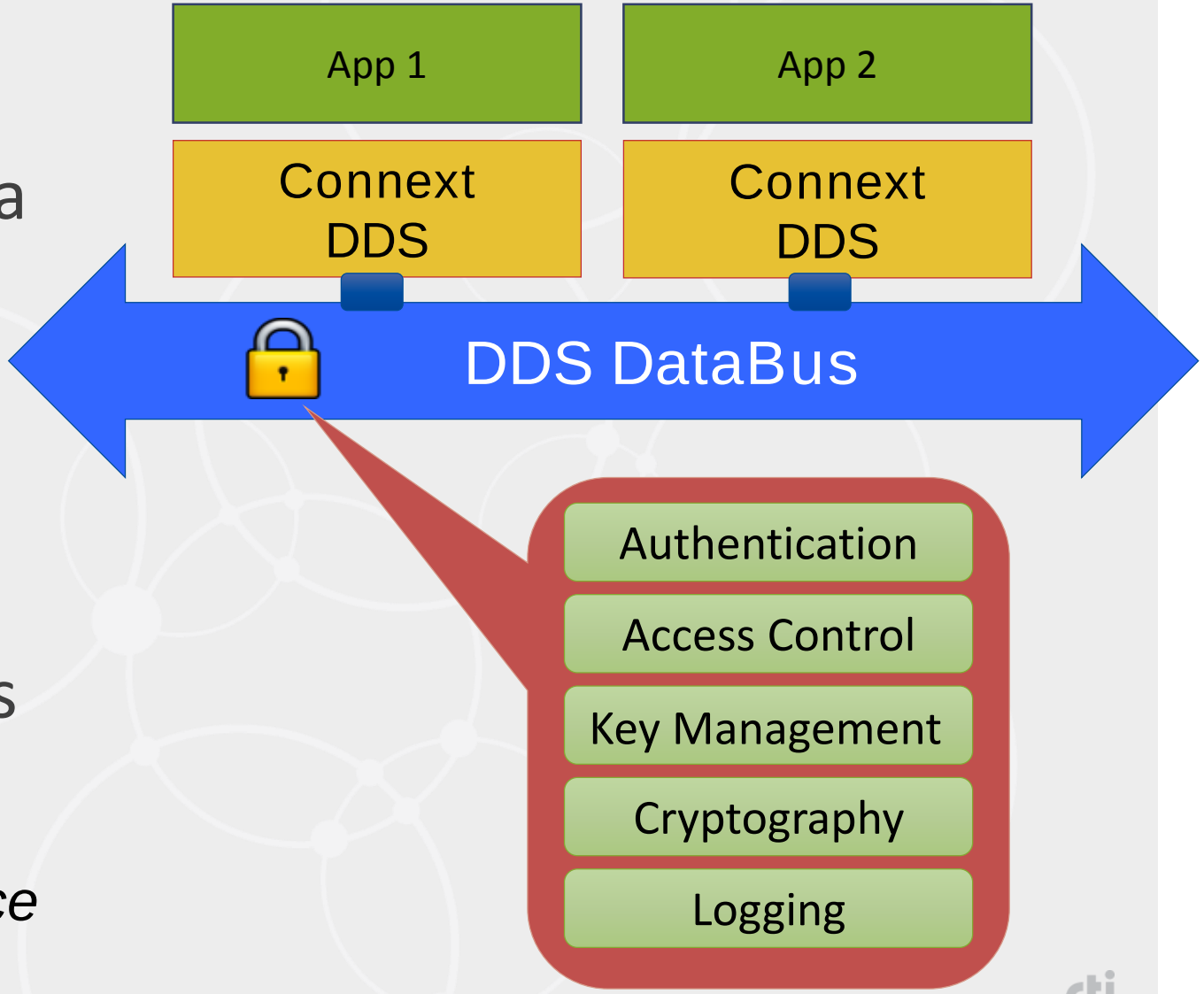
```
@mutable
struct ShapeType {
   @key string color;
   @range(0, 200)  long x;
   @range(0, 250)  long y;
   @optional @min(5) float size;
};

struct ShapeTypeExt : ShapeType {
    @unit("meter") long x;
};
```

```
/* Service definition */
enum Command { START, STOP };

@service
interface RobotControl
{
    void  command(Command com);
    float setSpeed(float speed)
          raises (TooFast);
     float getSpeed();
};
```

DDS Security Goals

- Authenticate subjects
- Enforce access control to data objects
- Ensure data integrity
- Ensure data confidentiality
- Enforce non-repudiation
- Provide availability of data
- Create auditable security logs

….while maintaining high performance



Complete C++ Example

// IDL

```
enum TempUnit {
    CELSIUS,
    FAHRENHEIT,
    KELVIN
};
struct TempType {
    @key short id;
    float     temp;
    TempUnit  unit;
};
```

// Publisher Application:

```
dds::domain::DomainParticipant dp(0);
dds::topic::Topic<TempType> topic(dp, "TTempSensor");
dds::pub::Publisher pub(dp);
dds::pub::DataWriter<TempType> writer(pub, topic);

TempType sensor(1, 0, 0, TempUnit::CELSIUS);

for ( int i = 0; i < 100; ++ i, ) {
   sensor.temp( i%100 );
   writer.write(sensor);

   std::this_thread::sleep_for(std::chrono::seconds(1));
}
```

// Subscriber Application:

```
dds::domain::DomainParticipant dp(0);
dds::topic::Topic<TempType> topic(dp, "TTempSensor");
dds::sub::Subscriber sub(dp);
dds::sub::DataReader<TempType> reader(sub, topic);

dds::sub::cond::ReadCondition condition(reader,
    dds::sub::status::DataState::any());
dds::core::cond::WaitSet waitset;
waitset += condition;

while (true) {
   waitset.wait(dds::core::Duration(4))
   auto samples = reader.take();
   for (auto s : samples) {
        std::cout << s.data() << std::endl;
   }
}
```

Connext DDS factsheet

- **Architecture:** Peer-to-Peer, no Broker, Layered (Hierarchical) Databus.
- **Communication Patterns:** Publish/Subscribe, Request/Reply, Queuing
- **Payload:** Strongly-defined types, opaque, mixed. Static/Dynamic.
- **Filtering:** Content filter, time filter, supports Publisher side filtering.
- **Quality of Service:** Extensive (Reliability, History, Liveliness, etc.)
- **Transports:** UDP (multicast), TCP, TLS, DTLS, shared memory, pluggable custom. Transparent Mobility.
- **Security:** Fine grained security per Topic, transport-level security.
- **Languages:** C, C++, Java, .NET, ADA. Via connector: JS, Python, Lua.

OPC UA Concepts

OPC UA Standards

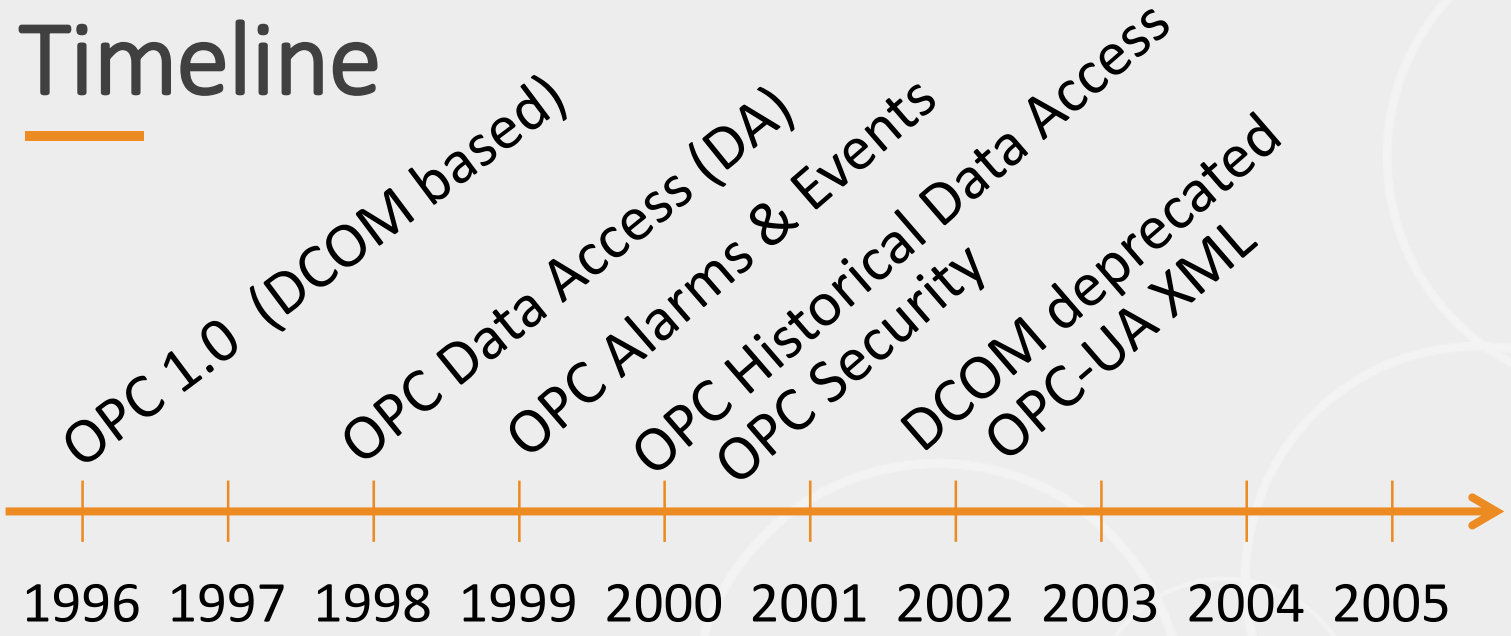
- Part I - Overview & Concepts
- Part 2 – Security Model
- Part 3 – Address Space Model
- Part 4 – Services
  - 5.1-5.3 – ...
  - 5.4 – Discovery
  - 5.5 – Secure Channel
  - 5.6 – Session
  - 5.7 – Node Management
  - 5.8 – View
  - 5.9 – Query
  - 5.10 – Attribute
  - 5.11 – Method
  - 5.12 – Monitored Item
  - 5.13 – Subscription
- Part 5 – Information Model
- Part 6 – Service Mapping
- Part 7 – Profiles
- Part 8 – Data Access
- Part 9 – Alarms & Conditions
- Part 10 – Programs
- Part 11 – Historical Access

Classic OPC

- Part 12 – Discovery
- Part 13 – Aggregates
- Part 14 – Publish-Subscribe

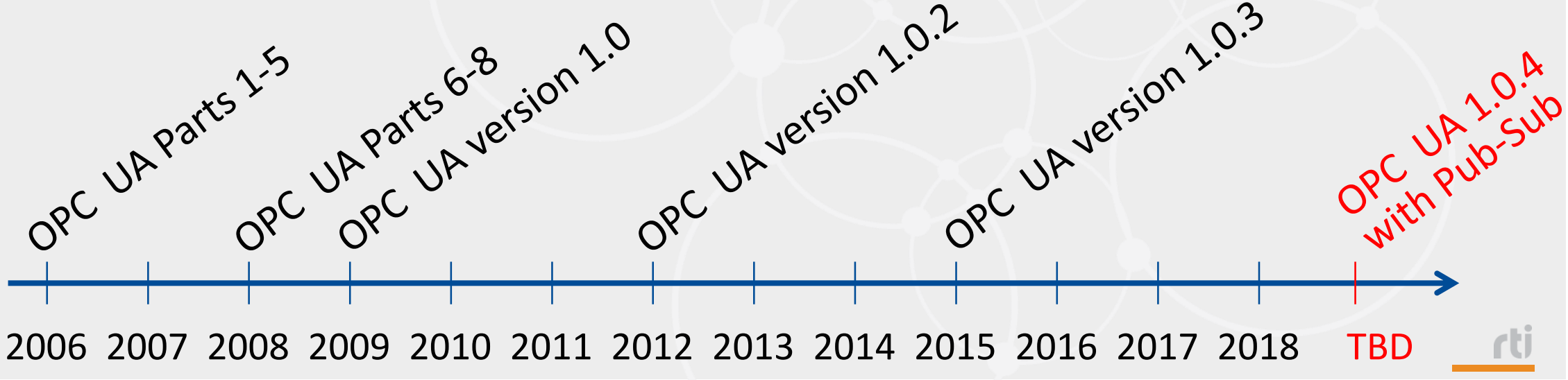
In Progress

Timeline



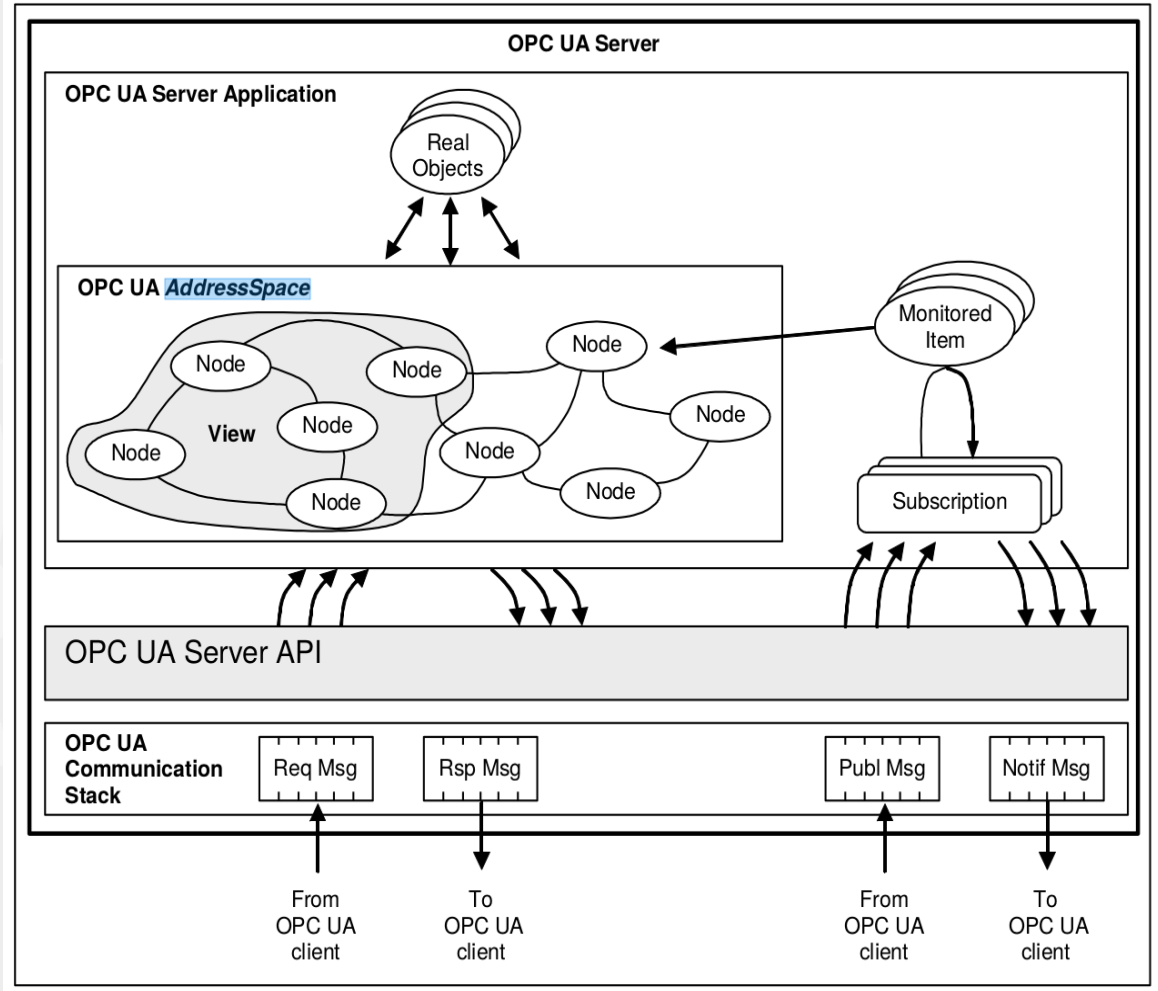
OPC Classic
(OLE for Process Control)

OPC UA (Open Platform Communication Unified Architecture)



OPC UA Basics

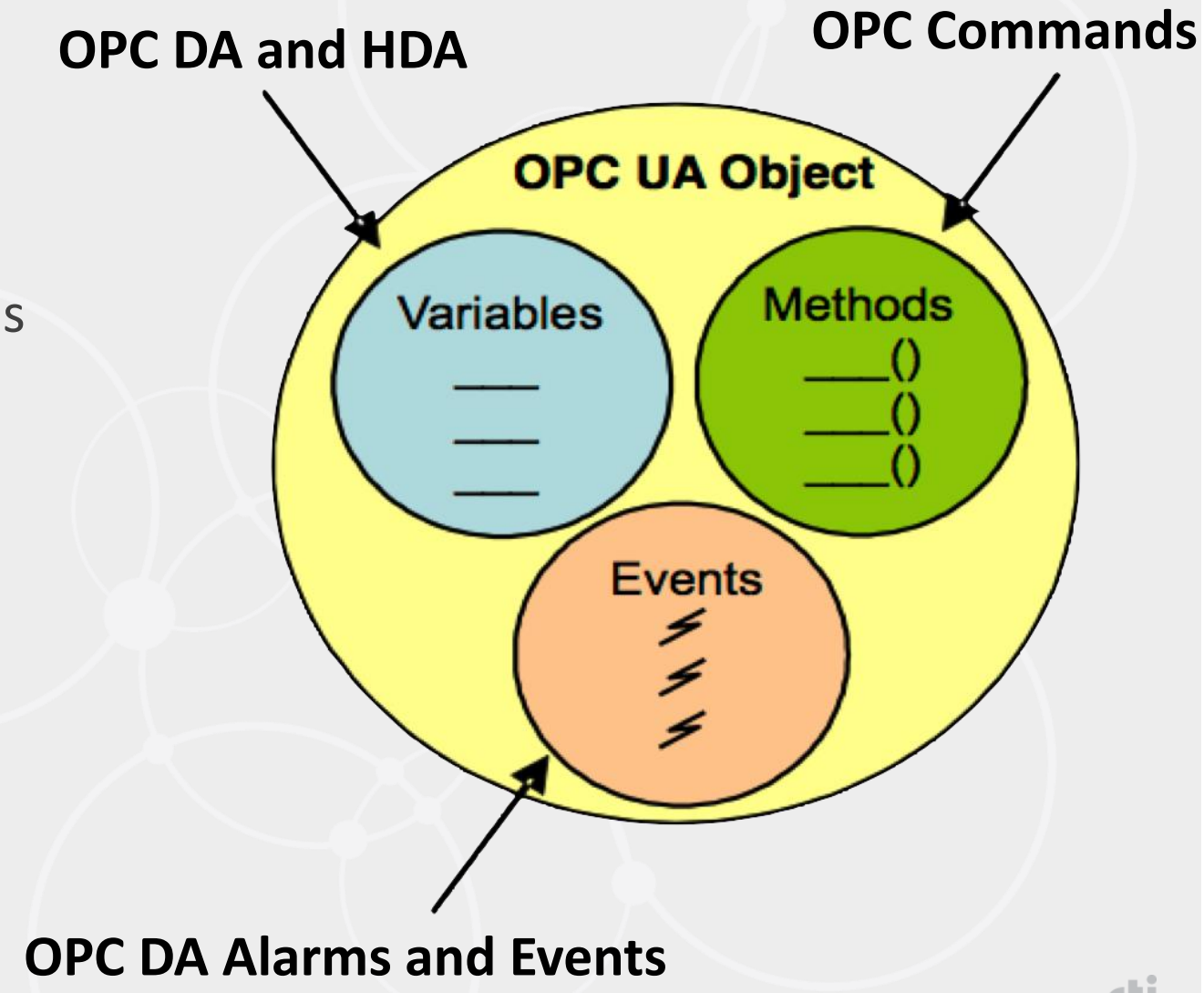
- Client/Server Model
  - Servers expose Address Space
  - Clients access information using services
- Object-oriented meta-model
  - Used for sensors, actuators…
  - Information as nodes in Address Space
  - Nodes are organized hierarchically
- OPC UA Services:
  - Connection management
  - Node management
  - View, Query
  - Attribute, Method
  - Subscription, Monitor items



OPC UA Basics

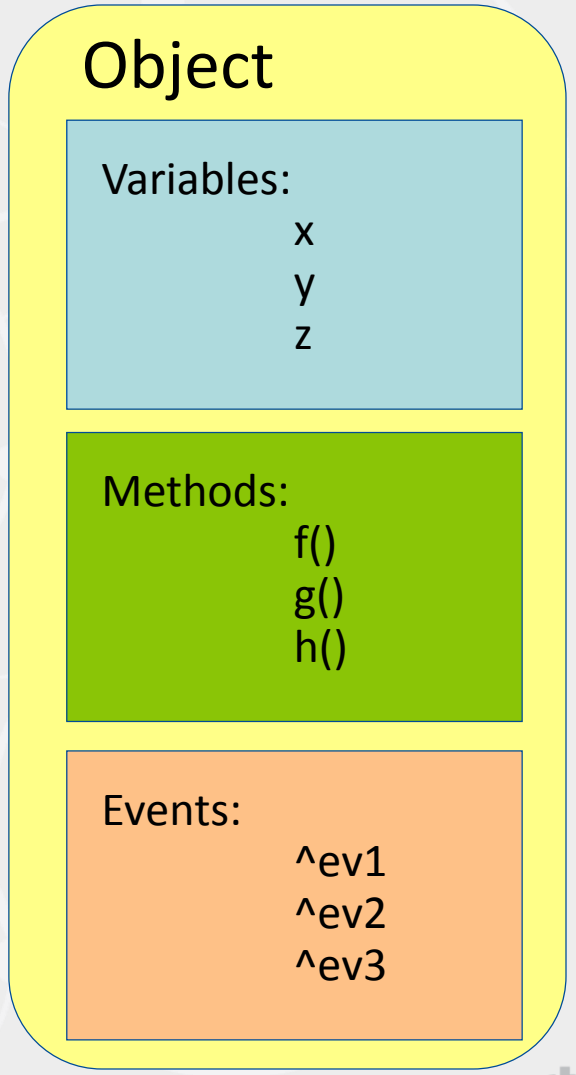
OPC UA = established OPC features

+ Platform independence
+ Standard internet and IP based protocols
+ Built-in security features
+ Generic object model
+ Extensible type system
+ Scalability through profiles
+ Migration path from Classic OPC



OPC UA Object model

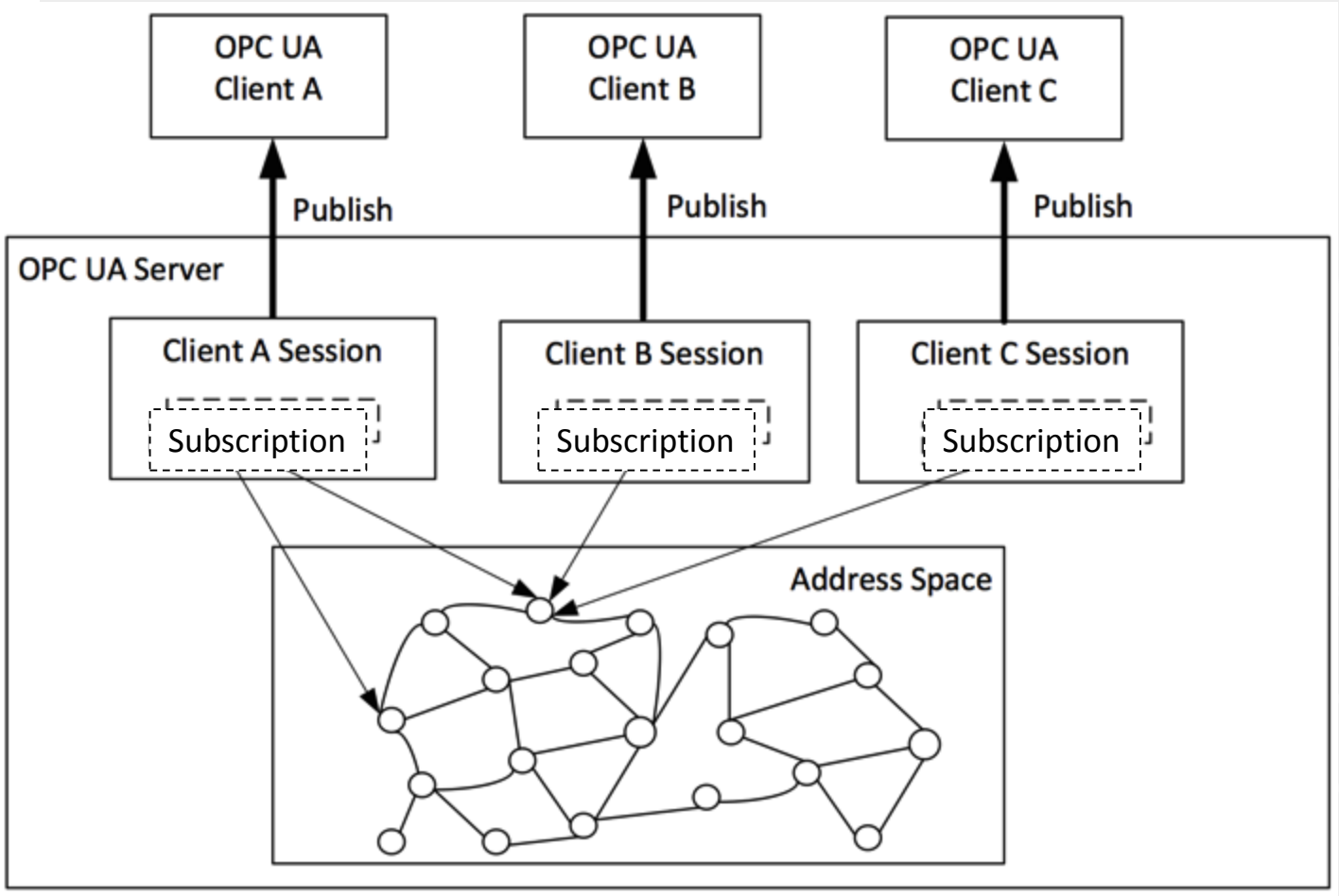
- Nodes and References between nodes
  - Nodes have: ***NodeId***, ***NodeClass***, ***BrowseName***, ***DisplayName***, ***Description***, ***WriteMask***, ***UserWriteMask***
  - References – relations between nodes
- Types of Nodes:
  - Object – Used to structure the address space. They group other Nodes. E.g. Variables and Methods always exists within the concept of an Object
  - Variable – represent a value Can be read, written, subscribed by a client
  - Method – Operations that may be called by a client, have arguments (input, output) and a result
  - Events – Notifications that may be subscribed by the client
  - Other: ReferenceType, VariableType, DataType, ObjectType, ...



Classic OPC UA Communication model

- Data Access
  - Client Browse:
    - Navigate Address space of sever
    - Create Variable Group. Read/Write group
  - Client Monitor
    - Create Variable Group & Monitor rate, Server push changes to any variables in group
- Alarms & Conditions
  - Client subscribes to “all notifications” from server that match a “filter criteria”
- Historical Data Access (HDA)
  - Read historical records. Three ways:
    - By time range, for a specific timestamp, as aggregated values
  - Insert, replace & delete data from the historical records

OPC UA Subscriptions



- Each *Client* creates **individual** *Sessions*, *Subscriptions* and *MonitoredItems* to:
  - Get *Events*
  - Observe *Variable Value* data changes
- *Subscriptions* are **not shared** between *Clients*

OPCUA Subscriptions vs DDS Subscriptions

- OPC UA “subscription”:
  - Lives on a specific OPCUA Server
  - Is created (dynamically) by a client with
    - Client-configured “refresh rate”
    - Type “sequence of variables” or “sequence of events”
  - Is sent **to just one client** containing:
    - Variables that changed in the last “refresh”
    - Events that occurred in the server

Unlike DDS, OPC UA Clients cannot share “subscriptions”:

- No global/shared data space. No databus.
- No multicast
- No Qos

OPC UA / DDS Gateway Standard

Date: February 2018

OBJECT MANAGEMENT GROUP

OPC UA/DDS Gateway

Version 1.0

OMG Document Number mars/2018-02-01

Normative Reference: http://www.omg.org/spec/DDS-OPCUA/1.0

Associated Normative Machine Consumable Files:

http://www.omg.org/spec/DDS-OPCUA/20180201/dds-opcua_builtin_types.idl
http://www.omg.org/spec/DDS-OPCUA/20180201/dds-opcua_services.idl
http://www.omg.org/spec/DDS-OPCUA/20180201/dds-opcua_subscriptions.idl
http://www.omg.org/spec/DDS-OPCUA/20180201/dds-opcua_definitions.xsd
http://www.omg.org/spec/DDS-OPCUA/20180201/dds-opcua_definitions_nonamespace.xsd
http://www.omg.org/spec/DDS-OPCUA/20180201/dds-opcua_model.xmi

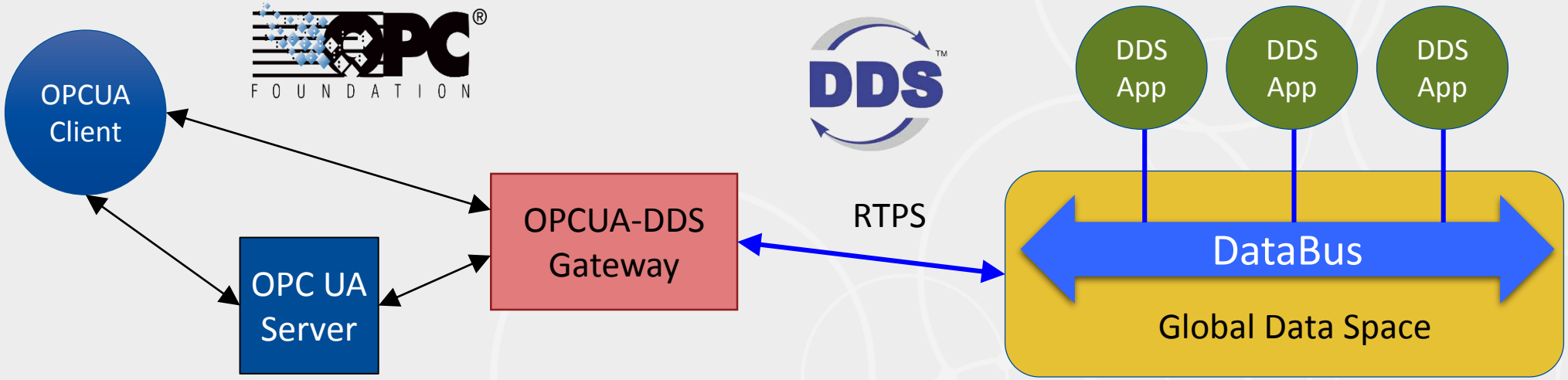
Associated Non-Normative Machine Consumable Files:

http://www.omg.org/spec/DDS-OPCUA/20180201/dds-opcua_dds2opcua_configuration.xml
http://www.omg.org/spec/DDS-OPCUA/20180201/dds-opcua_opcua2dds_configuration.xml

---

OMG Specification approved March 2018

OPC UA / DDS Gateway



- Existing: OPC UA Server(s)
- Existing: OPC UA Client application(s)
- Existing: DDS Application(s)
- New: OPCUA-DDS Gateway– bridges between OPCUA and DDS

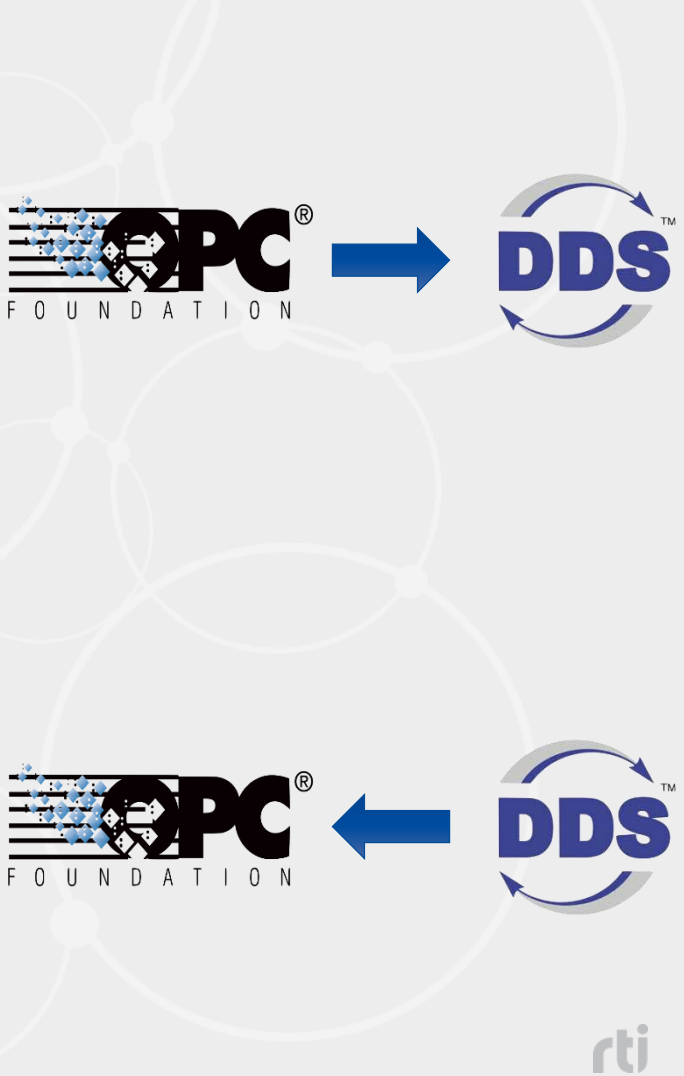
Specification Overview

The specification defines 5 building blocks that build up the OPC UA/DDS Gateway:

- OPC UA Type System Mapping
- OPC UA Service Sets Mapping
- OPC UA Subscription Model Mapping
- DDS Type System Mapping
- DDS Global Data Space Mapping

Four Conformance Points

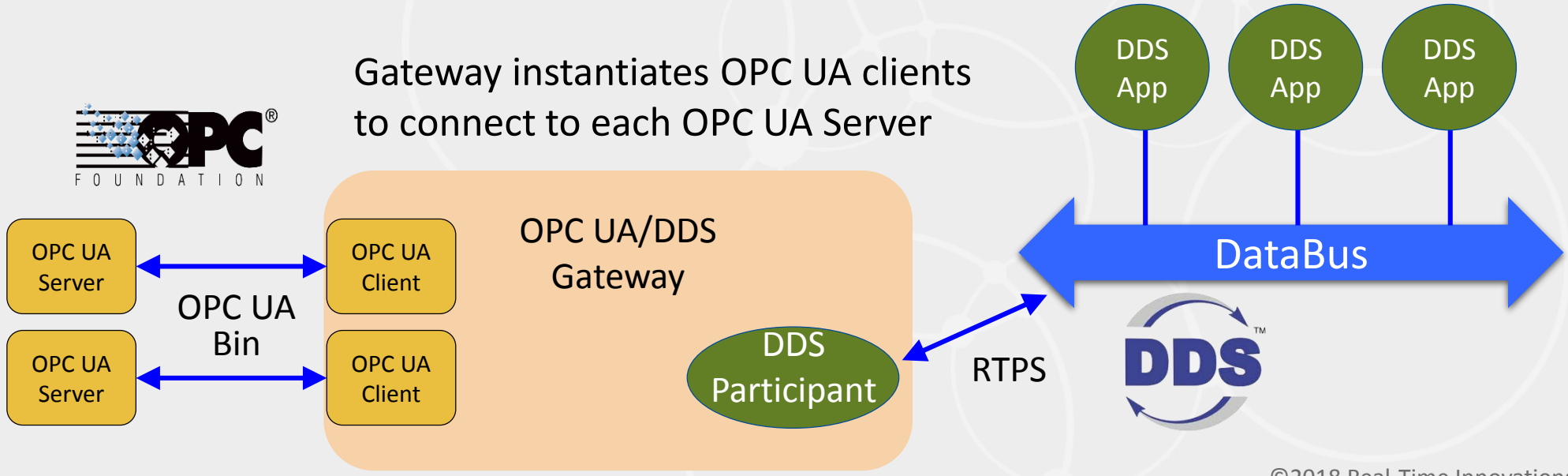
- OPC UA to DDS / Basic Conformance
  - OPC UA Subscription Model Mapping
  - OPC UA Type System Mapping
- OPC UA to DDS / Complete Mapping
  - Basic Conformance + OPC UA Service Sets Mapping
- DDS to OPC UA / Basic Conformance
  - DDS Global Data Space Mapping (except HDA, 9.3.4.4)
  - DDS Type System Mapping
- DDS to OPC UA / Complete Conformance
  - Basic Conformance + HDA (9.3.4.4)



OPC UA to DDS

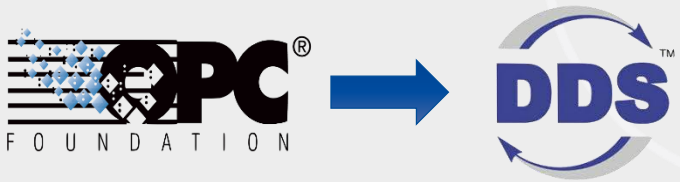
OPC UA to DDS—Overview

- Building blocks:
  - OPC UA Type System Mapping to DDS
  - OPC UA Service Invocation from DDS
  - OPC UA Subscription from DDS



OPC UA Type System Mapping—Primitive Types

- OPC UA primitive types map easily to DDS/IDL types



OPC UA Type	DDS Type (IDL 4.2)
Boolean	`boolean`
SByte	`int8`
Byte	`uint8`
Int16	`int16`
UInt16	`uint16`
Int32	`int32`
UInt32	`uint32`
Int64	`int64`
UInt64	`uint64`
Float	`float`
Double	`double`
String	`string`

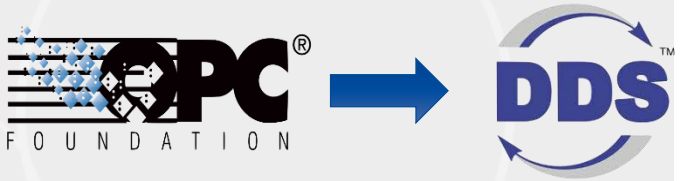
OPC UA complex types mapped to DDS structures

Examples:

```
struct Guid {
    uint32 data1;
    uint16 data2;
    uint16 data3;
    octet  data4[8];
};
```

```
struct NodeId {
    uint16 namespace_index;
    NodeIdentifierType identifier;
};
```

```
union NodeIdentifierType switch(NodeIdentifierKind) {
  case NODEID_NUMERIC:
    uint32 ulong_val;
  case NODEID_STRING
    string str_val;
  case NODEID_GUID:
    Guid   guid_val;
  case NODEID_OPAQUE:
    sequence<octet> opaque_val
};
```



OPC UA Type System Mapping—Variants

```
union VariantValue
switch(BuiltinTypeKind) {
  case BOOL_TYPE:
    boolean bool_val;
  case SBYTE_TYPE:
    int8 sbyte_val;
  case BYTE_TYPE:
    uint8 byte_val;
...
};

struct Variant {
    sequence<uint32> array_dimensions;
    sequence<VariantValue> value;
};
```

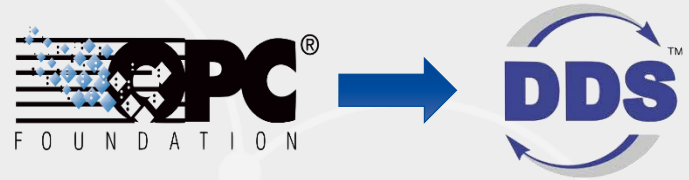
OPC UA Monitored Items & Subscription Mapping

- Monitored Items and Subscriptions are mapped to a DDS Topic
  - DDS Applications can subscribe to the Topic on the DDS
  - The Gateway manages everything:
    - Connecting to the OPC UA server
    - Setting up the monitored items & subscription
    - Publishing the OPC Server data to DDS, including type mapping
- The Gateway makes OPC UA Subscriptions scalable
  - Unlike OPC UA in DDS many Subscribers can receive data from the same Publisher

OPC UA to DDS— Subscription Configuration

```
<dds>
  <types>...</types>
  <ddsopcua_gateway name="...">
    <opcua_connection name="..." server_endpoint_url="...">...</opcua_connection>
    <domain_participant name="..." domain_id="...">...</domain_participant>
    <opcua_to_dds_bridge name="...">
      <opcua_input name="..." opcua_connection_ref="...">
        <subscription_protocol>...</subscription_protocol>
          <monitored_items>
            <data_item name="..."></data_item>
            <event_item name="..."></event_item>
          </monitored_items>
        </opcua_input>
      <dds_output name="..." domain_participant_ref="...">...</dds_output>
      <mapping>
        <assignment dds_output_ref="..." opcua_input_ref="...">...</assignment>
      </mapping>
    </opcua_to_dds_bridge>
  </ddsopcua_gateway>
</dds>
```

OPC UA Service Sets Mapping



- OPC UA services are a collection of remote procedure calls that OPC UA Clients can invoke on the OPC UA Servers
- They are mapped to DDS services using DDS-RPC

Service Sets

IDL-defined DDS Services

Example: OPC UA Attributes Service Set

Attribute Service Set

- Access to node attributes (e.g., read the values)
- DDS applications may use these interfaces to access attributes of OPC UA object.

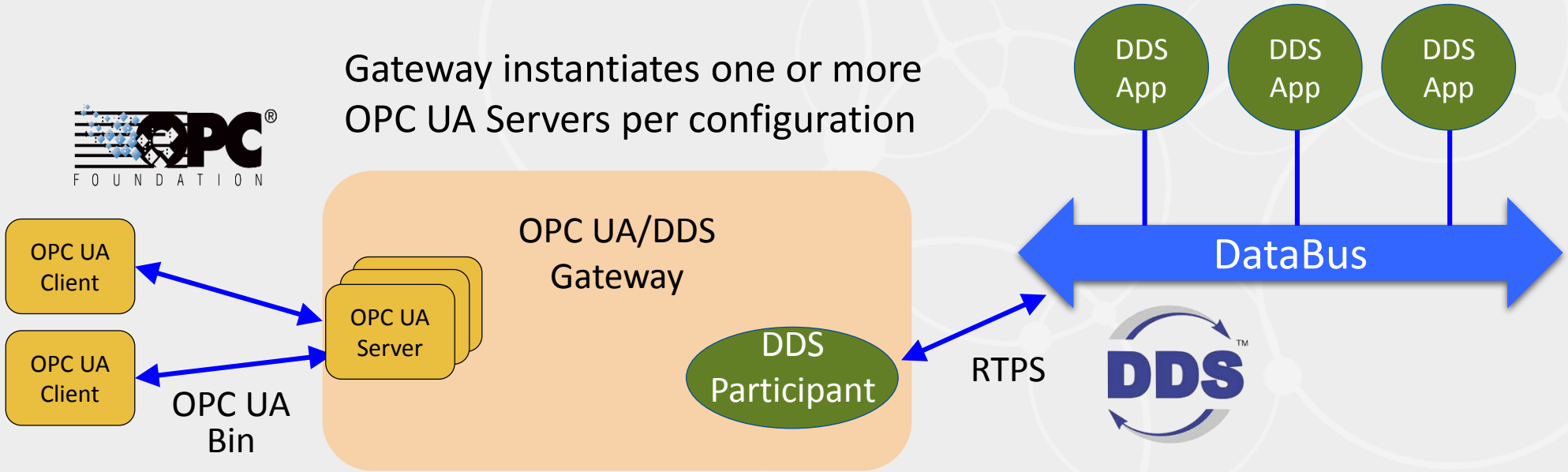
```
@DDSService
interface Attribute {
  ResponseHeader read(
    in  string server_uri, // Identifies OPC server
    out sequence<DataValue> results,
    out sequence<DiagnosticInfo> diagnostic_infos,
    in  Duration max_age,
    in  TimestamsToReturn timestamps_to_return,
    in  sequence<ReadValueId> nodes_to_read);

  ResponseHeader write(
    in  string server_uri, // Identifies OPC server
    out sequence<StatusCode> results,
    out sequence<DiagnosticInfo> diagnostic_infos,
    in  sequence<WriteValue> nodes_to_write);

    ...
};
```

DDS to OPC UA

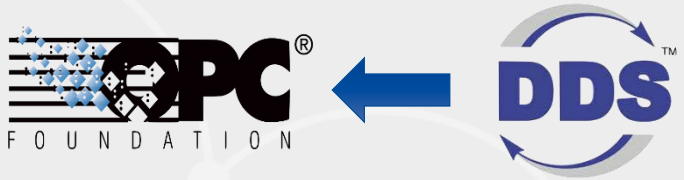
DDS to OPC UA—Overview

Building blocks:

- DDS Type System Mapping to OPC UA
- DDS Global Data Space Representation in OPC UA



Type System mapping

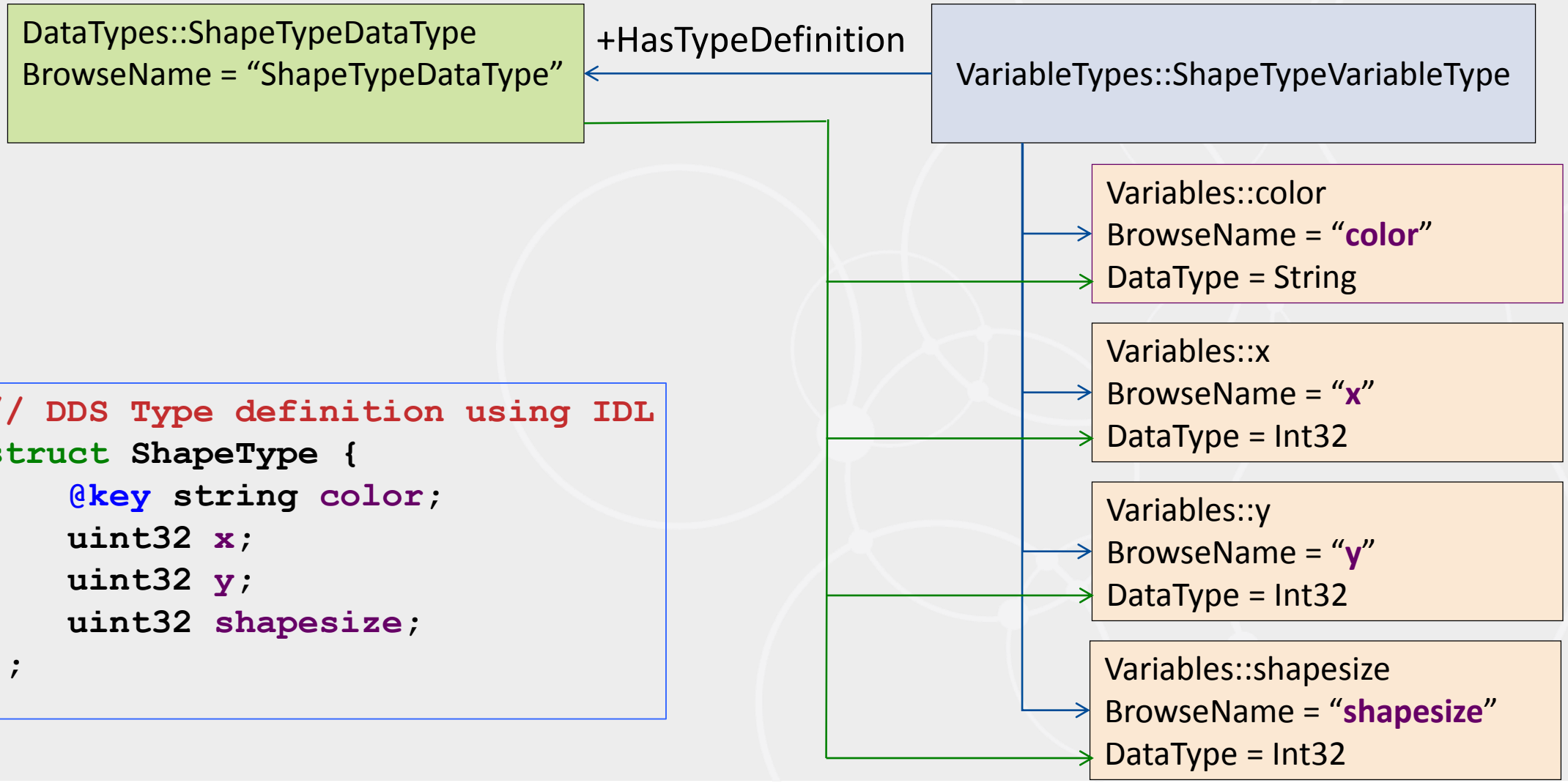


- DDS primitive and string types are exposed as OPC UA Nodes in the Gateway OPC UA Server
- DDS Union Types are exposed to OPC UA VariableType Union containing Variables of each specific discriminator value
- DDS Structured Types are exposed in two ways:
  - As as OPC UA Structure DataType (ExtensionObject)
    - Provides access to all fields as a whole in serialized form
  - As as OPC UA Complex VariableType containing Variables for each field
    - Provides convenient individual access to each field that can be processed by Generic OPC UA Clients
- DDS Array and Sequence types are exposed two ways:
  - As a whole
  - As a OPC UA VariableType Object with nested Variables for each element with names automatically created from the structure name and index

Example: Mapping of struct “ShapeType”

OPC UA representation as *Structure DataType*

OPC UA representation as complex *VariableType*



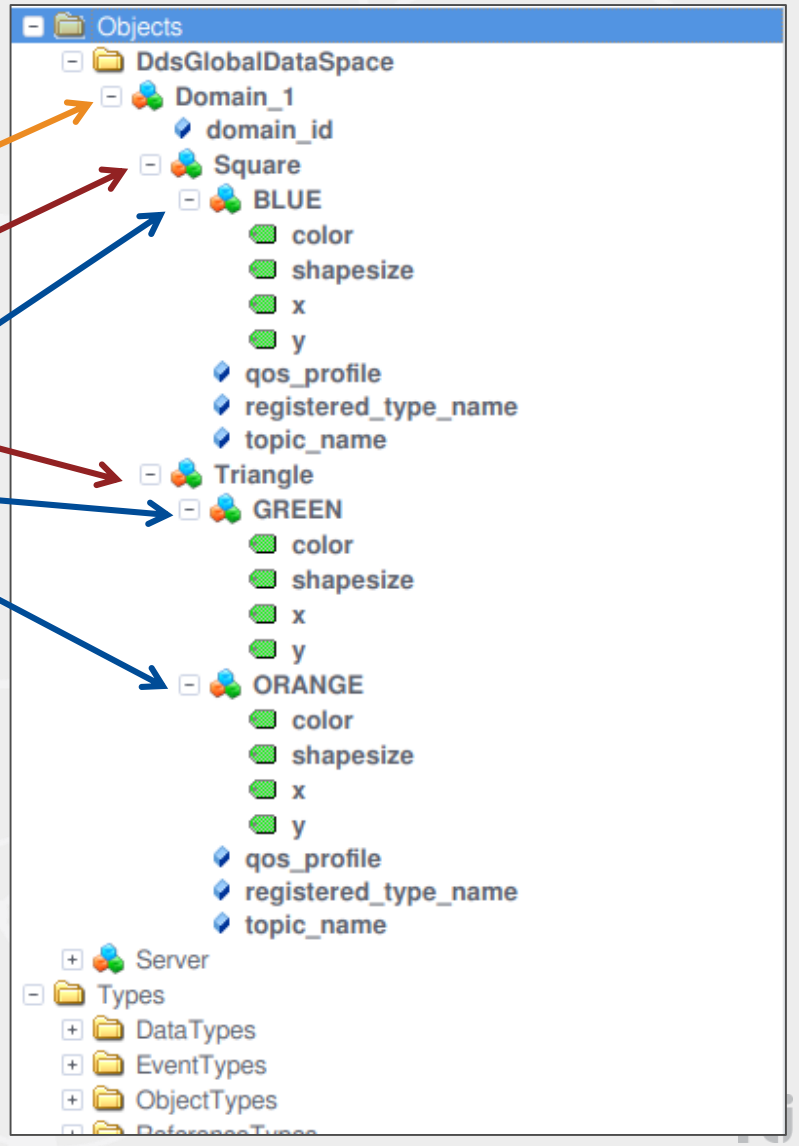

```
// DDS Type definition using IDL
struct ShapeType {
    @key string color;
    uint32 x;
    uint32 y;
    uint32 shapesize;
};
```

DDS Global Data Space Representation

DDS Global Data Space elements:

- Domain
  - Topic
    - Topic Instances (Keys)

are exposed as *Nodes* in the Address Space of the Gateway OPC UA Server

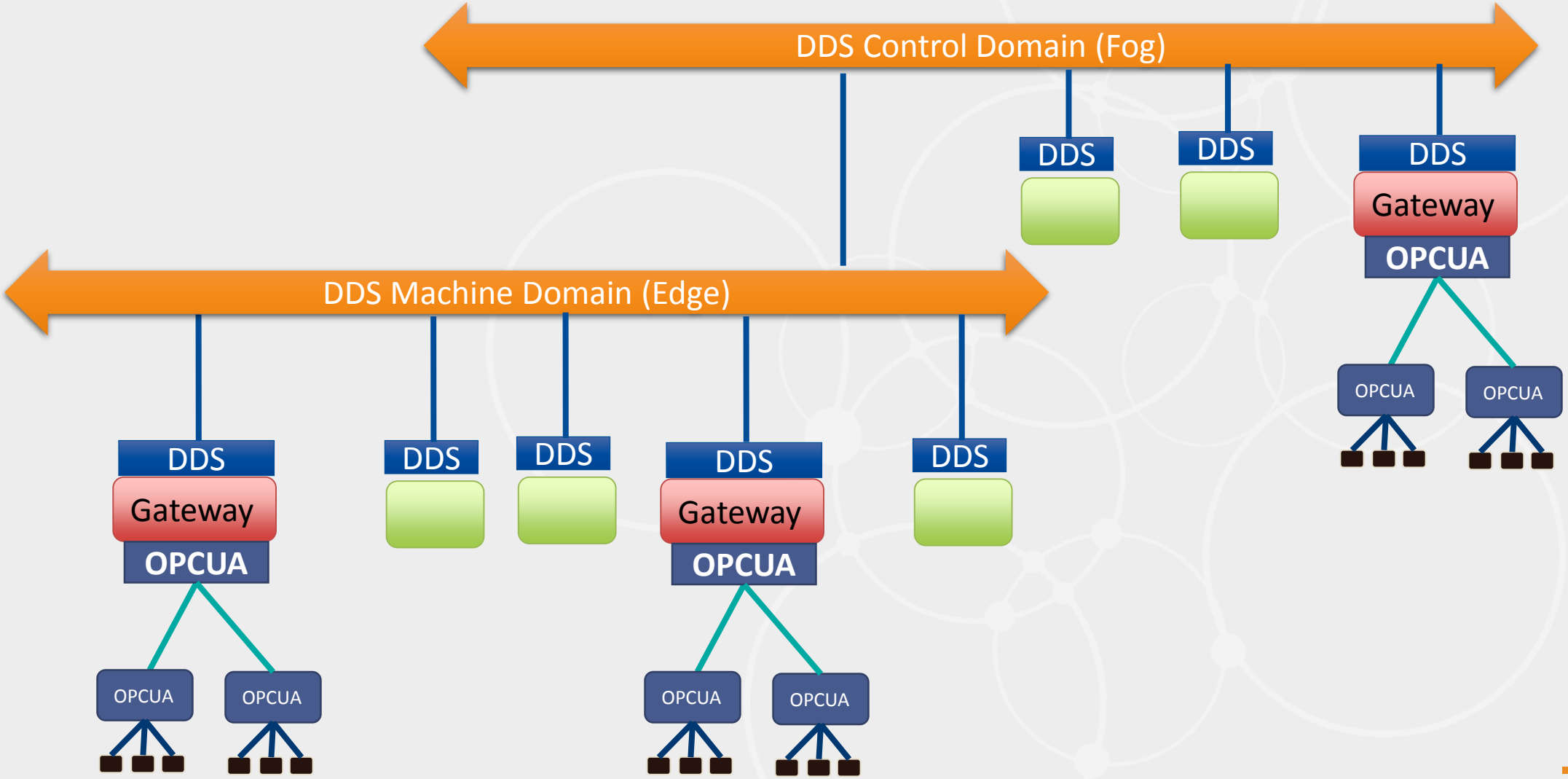


Domain with domain_id=0 has two DDS Topics: “Square” and “Circle”

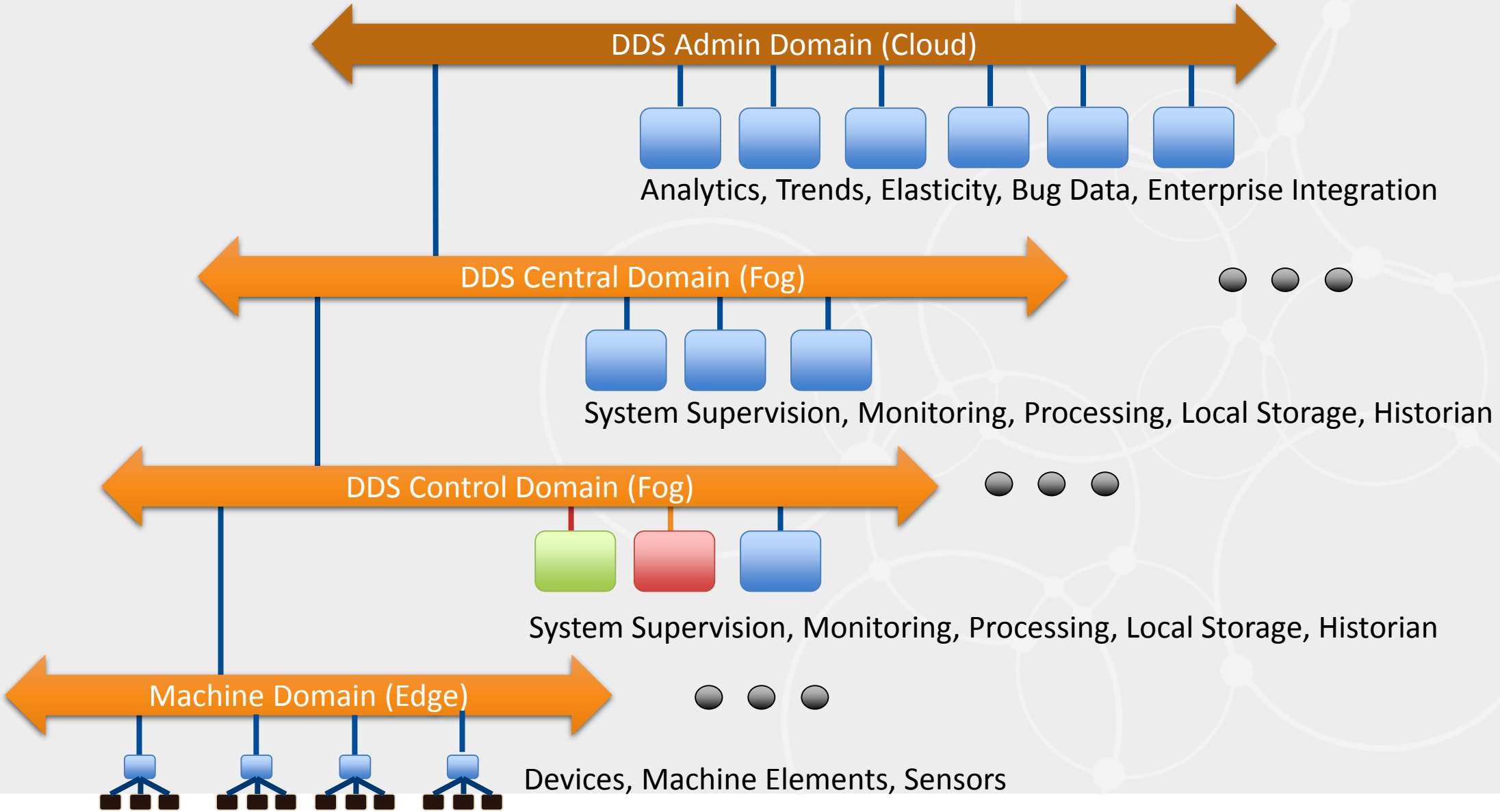
- Topic “Square” has instance “BLUE”
- Topic “Triangle” has instances “GREEN” and “ORANGE”

Conclusion

Edge / Machine Domain



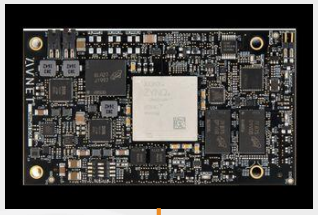
Scalable Architecture Edge to Cloud



DDS/OPC-UA Prototype and Demo

Codesys Control
OPC UA Server
POWERLINK MN

UltraZed-EG SOM



OPC-UA to DDS Gateway
Mocana Security
TPM 2.0

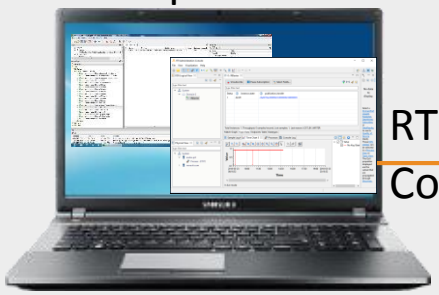
Edge Node
HW TBD (SOC-E?)

DDS Endpoint

3.3V, GND,
Signal (IN)
GPIO Pin TBD

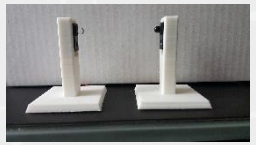
3.3V, GND,
Signal (OUT)
GPIO Pin 417

UaExpert



RTI Admin Console

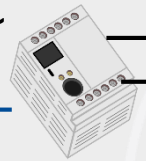
Switch



IR Barrier

Switch

PLC

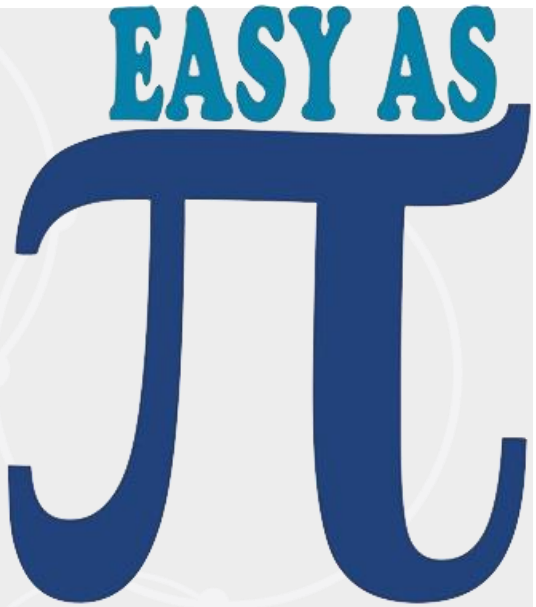


Motor

Ethernet
Powerlink
GPIO
Analog

https://www.rti.com/blog/announcing-the-opc-ua-dds-gateway-standard

Simplicity, Scalability, Performance



- Normally accessing OPC UA data is quite complex…
  - Define subscription
  - Define Monitored Items
  - Read data as an array of “Variants,” address via indices
  - Use untyped API’s…
- With the DDS Gateway it is simple!
  - Define the DDS data-types in XML
  - List and map the OPC UA monitored items in the XML file
  - Use DDS API to subscribe to the Topic…

And added benefits of DDS Performance, Scalability and QoS

Thank You!

Audience Q & A

Dr. Gerardo Pardo-Castellote,
Chief Technology Officer,
RTI

OpenSystems media

Thanks for joining us

Event archive available at:

http://ecast.opensystemsmedia.com/

E-mail us at: jgilmore@opensystemsmedia.com