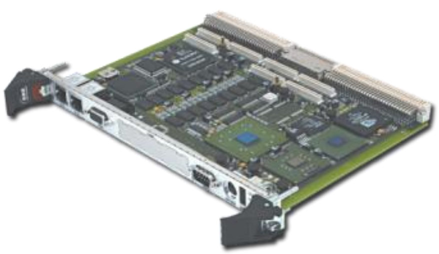
TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

Twin Oaks Computing, Inc.

How to Connect the Diverse Platforms of the IIoT

June 2015

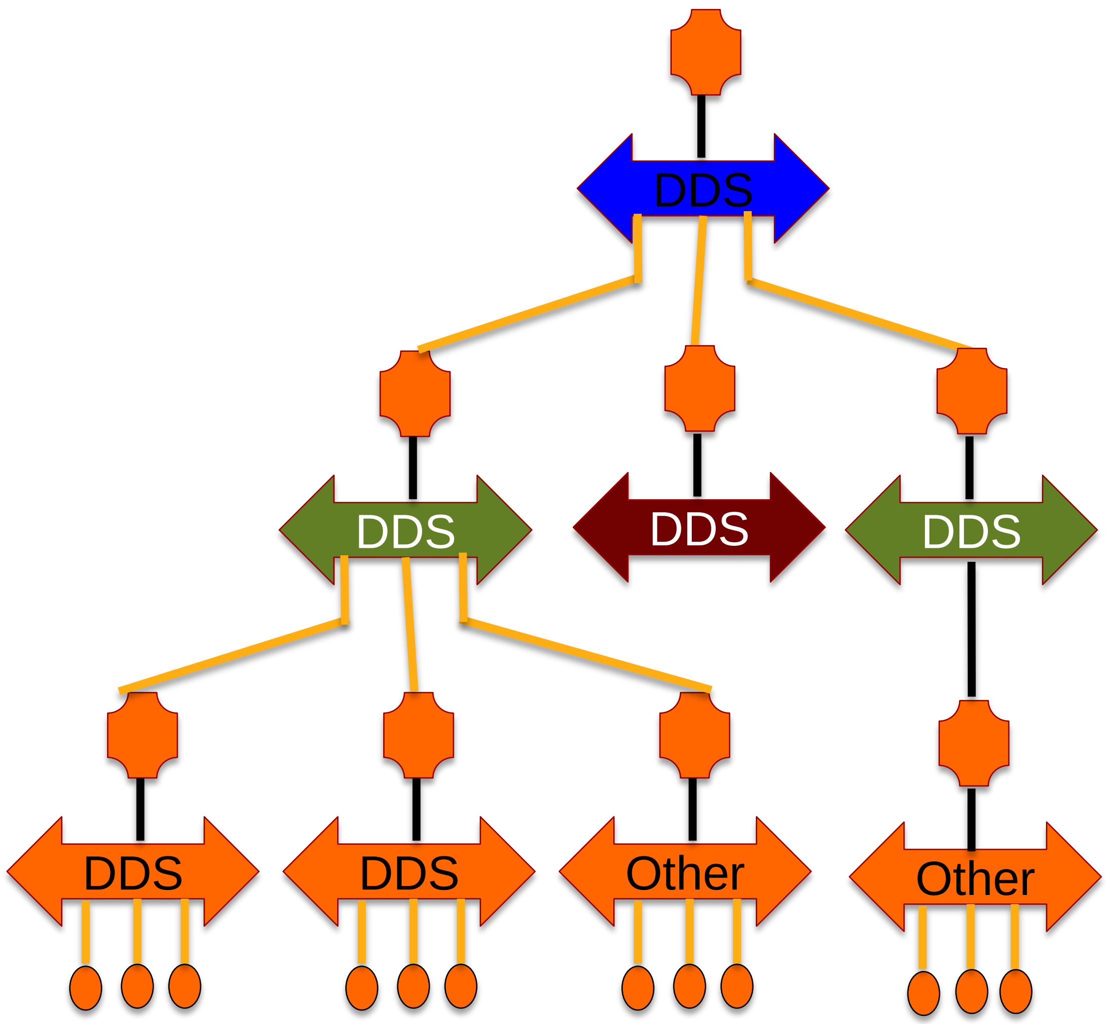


CoreDX™ DDS

Introduction & Agenda

- Nina Tucker, VP Twin Oaks Computing, Inc.
- DDS Platform Support, Interoperability
- DDS Protocol, Features
- DDS Performance, Scalability
- DDS and other IoT middleware technologies

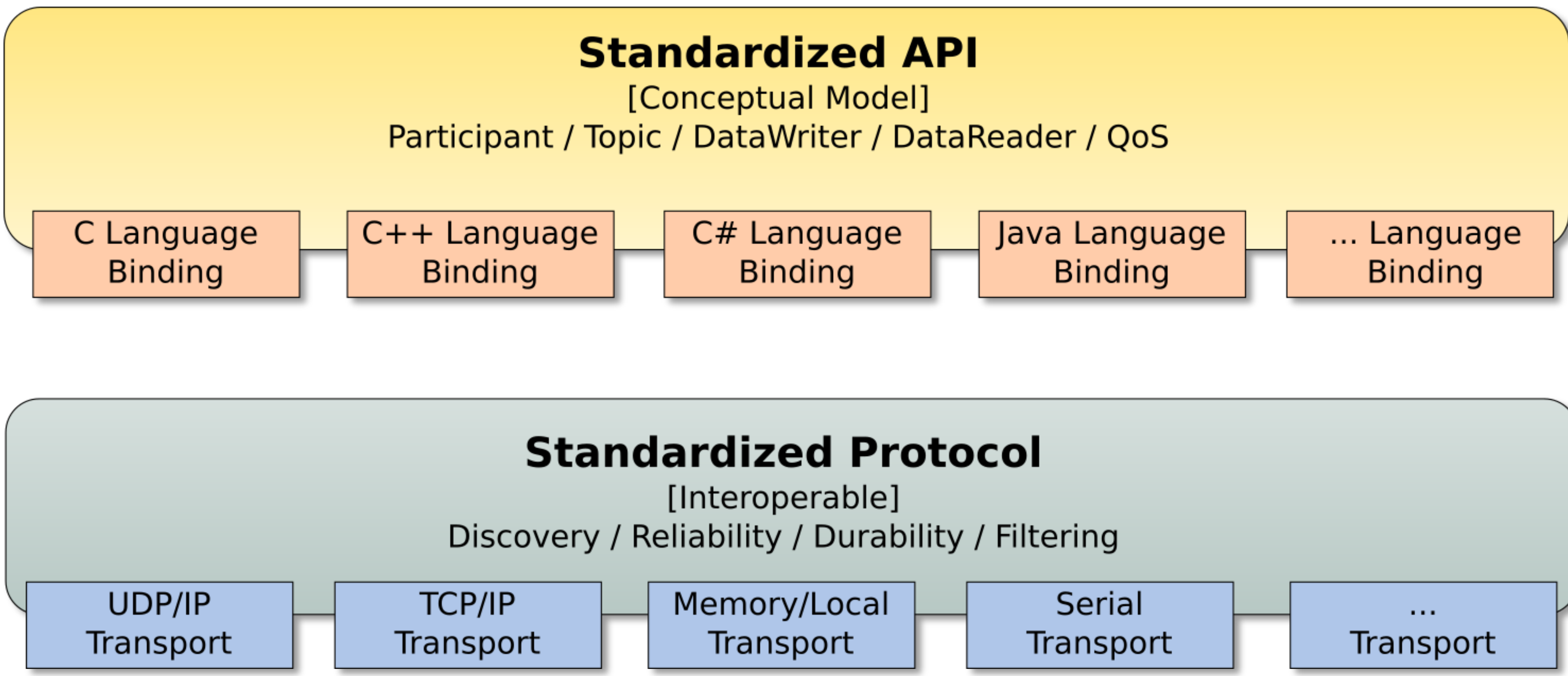
Typical IIoT Architecture



- Cloud:
  - Datacenter
  - Elasticity, Provisioning, Management
- Fog:
  - Distributed computing
  - Processing “close to the edge”
  - Latency, Robustness, availability
- Edge:
  - Locality
  - Information Scoping

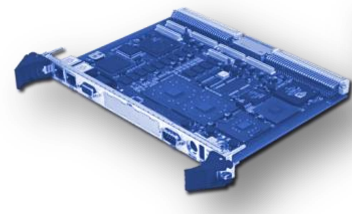
DDS Platform Support and Interoperability

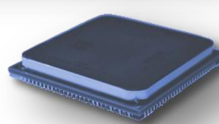
DDS Platform Support
Portable, Interoperable



DDS Platform Support:
Language, Platform, OS Support

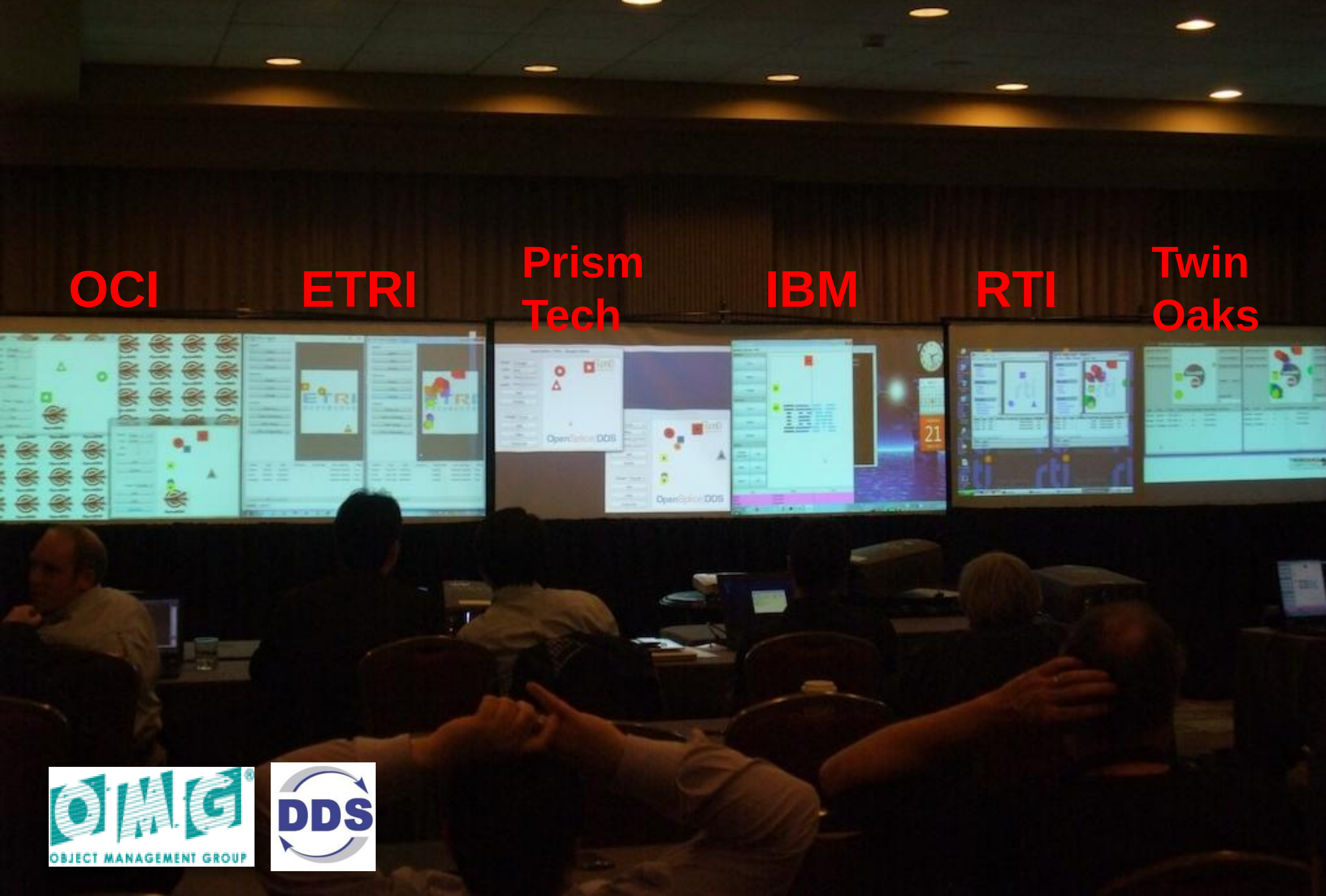
- Languages:
  - C, C++, C#, Java, Ada, Perl, Scala
- Hardware Architectures:
  - x86 (32 & 64 bit), UltraSPARC, ARMv5, ARVMv7, PPC, MIPS, Microblaze, FPGAs, DSPs, PLCs
- Operating Systems:
  - Linux, Windows, Mac OSX, Solaris, QNX, VxWorks, NexusWare, LynxOS, INTEGRITY, Android, iOS, WinCE, Unison, ThreadX
- Transports:
  - UDP (Interoperable), TCP, Shared Memory, Local Machine, SSL, Serial, Zigbee, Backplane, RDMA





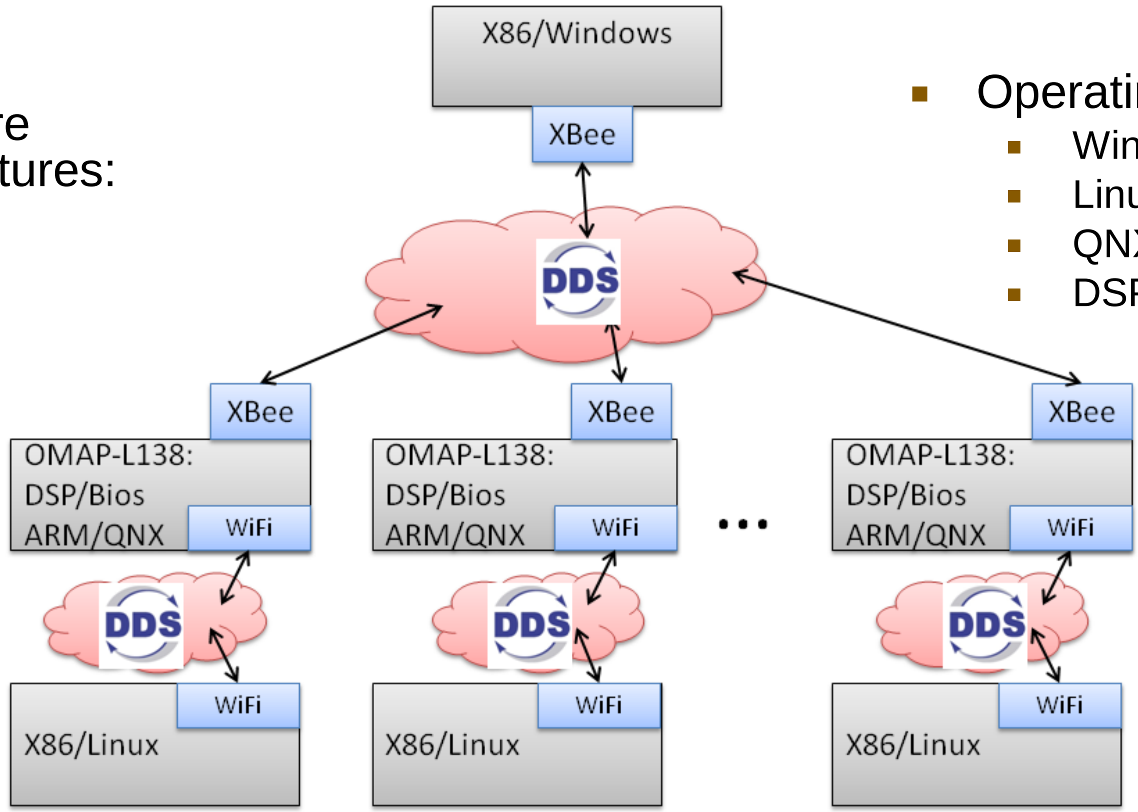
TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

DDS Platform Support: Protocol Interoperability



Case Study:
Wearable Sensors & Mobile Relays

- Hardware Architectures:
  - x86
  - arm
  - DSP



- Operating Systems:
  - Windows
  - Linux
  - QNX
  - DSP Bios

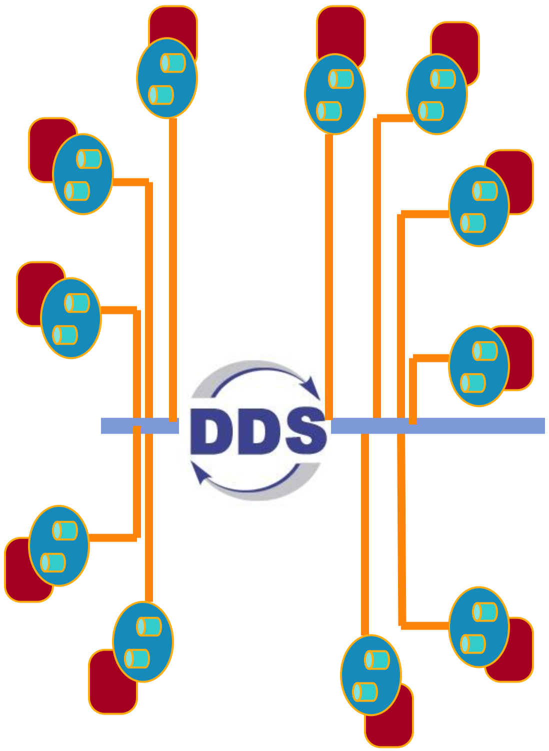
- Transports:
  - UDP (Wifi, Wired)
  - Xbee

TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

DDS Protocol and Features

DDS Protocol:
Features Optimized for IIoT

- Peer to peer no brokers or servers
- Adaptable QoS, including prioritization
- Reliable even over unreliable transports
- Any size data automatic fragmentation
- Automatic Discovery and Presence without configuration
- Decoupled execution start/stop apps in any order
- Redundant sources, sinks, paths, networks
- Efficient use of resources (network, CPU, memory)
- High performance near-native “wire” speeds
- Scalable no N^2 network connections
- Secure end-to-end communications



DDS Protocol Case Study:
Medical Device Domain

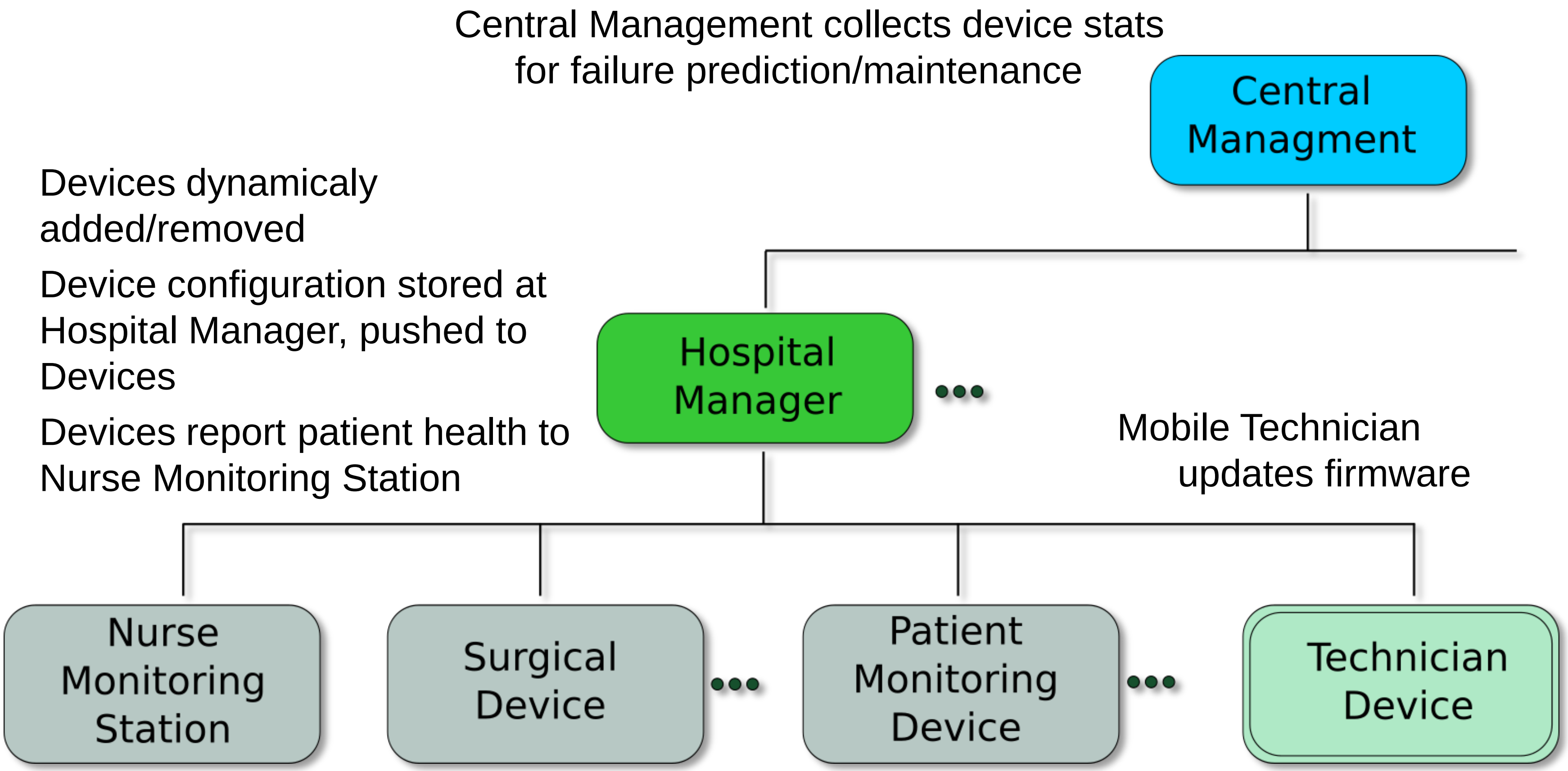
- Communication Requirements:
  - Components on **FPGA's** (embedded **RTOS**), Atoms (**Embedded Linux**), Desktop (**Windows**), Tablets (**Android, iOS**)
  - Networks: directly connected wired Ethernet, local area WiFi, Internet
  - Some very **low latency** requirements, some **secure** communication requirements, over different networks

- Benefits provided by DDS:
  - Code reuse among devices
  - Common API for sending and receiving data between distributed device components
  - Flexible architecture: ability to move software components to different devices late in the development cycle without schedule impacts
  - Ability to quickly and easily create test programs and emulators to emulate hardware components not yet available to developers (buttons, switches, lights on medical device)

TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

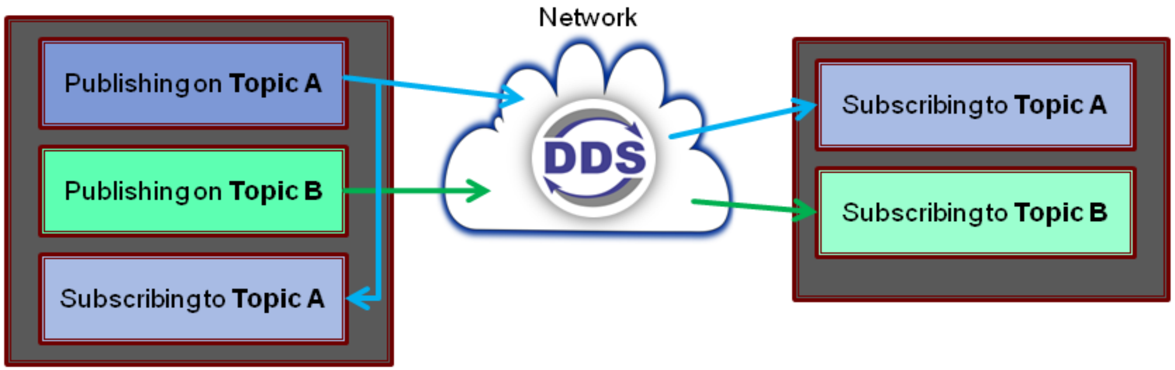
DDS Protocol Case Study:
Medical Device Domain

- Devices dynamicaly added/removed
- Device configuration stored at Hospital Manager, pushed to Devices
- Devices report patient health to Nurse Monitoring Station



DDS Protocol: Discovery and Liveliness

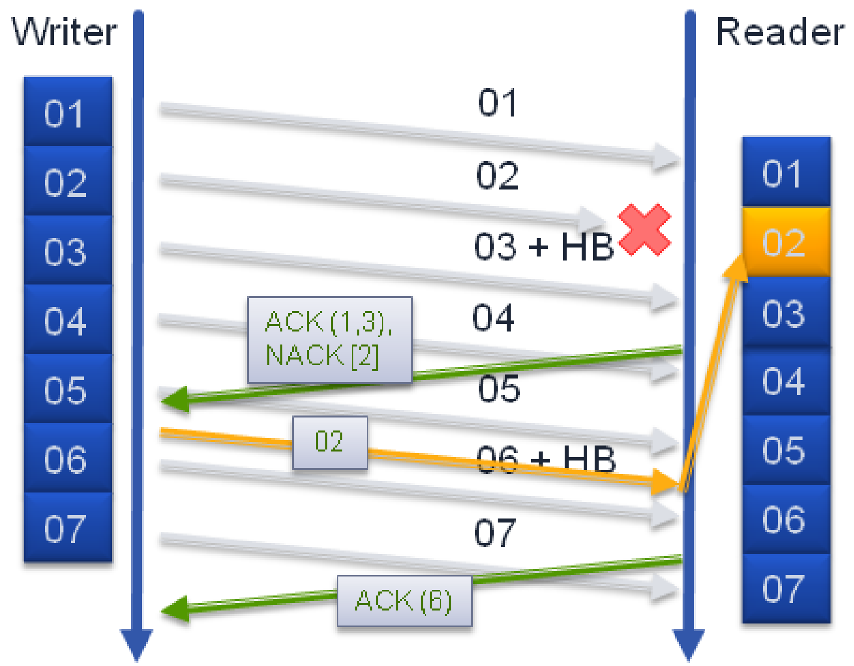
- DDS provides Dynamic Discovery of publishers and subscribers
  - Application does not know or configure DDS endpoints: they can be on the same machine or across a network
  - Simply indicate intent to publish or subscribe to data and let DDS do the rest



- DDS Liveliness manages existence of peers
  - Exiting cleanly or not
  - Coming back online

DDS Protocol:
Reliability

- Configurable **Reliable** data communications
  - Even on unreliable and intermittent networks
  - Using UDP (MULTICAST and UNICAST)
- DDS Reliability protocol Addresses
  - Dropped Packets
  - Out of Order Samples
  - Communication Disconnects
  - Application Re-Starts



DDS Protocol:
Durability

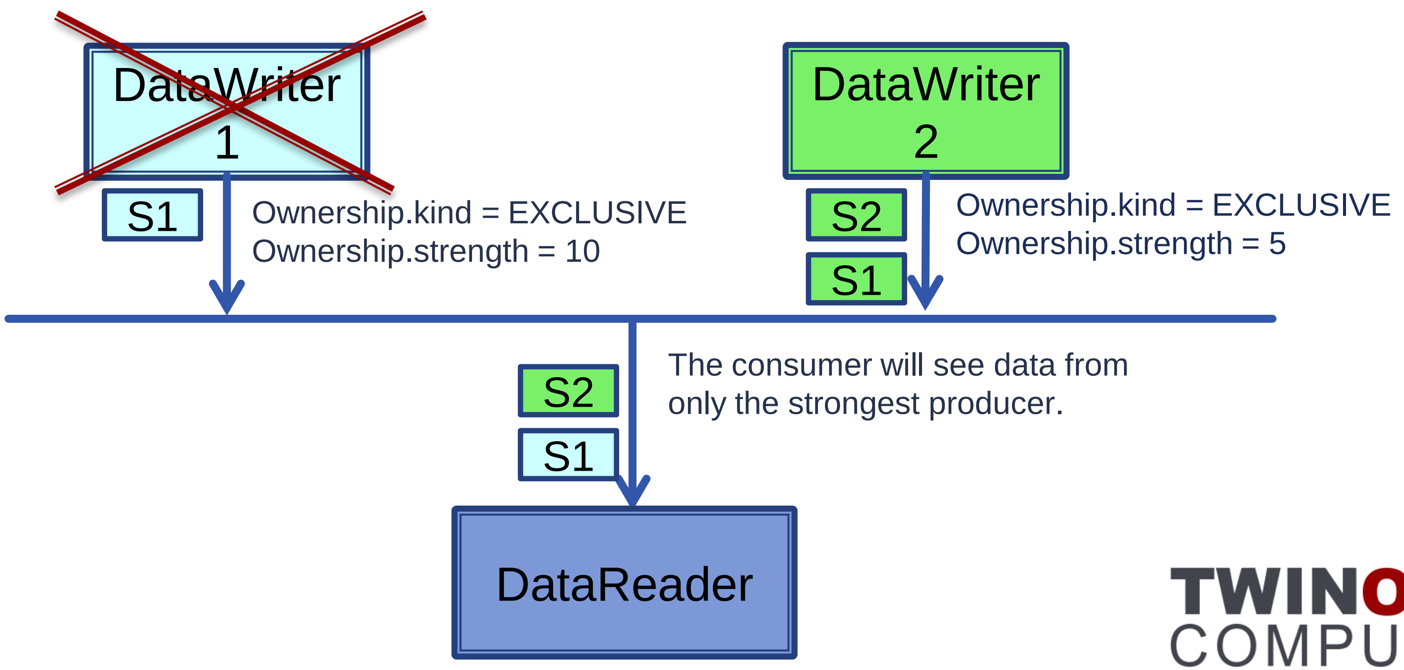
- DDS provides configurable **Durable** data communications

- DDS Durability addresses
  - How long data is published saved
  - If previously published data is sent to ‘late joining’ DataReaders

- Durability Options
  - Volatile: Published data is saved just long enough to be sent to currently matched Readers
  - Transient Local: Published data is saved for as long as the Data Writer is alive (until application exit)
  - Transient, Persistent: Published data is saved by an external service, through application restarts, machine reboots

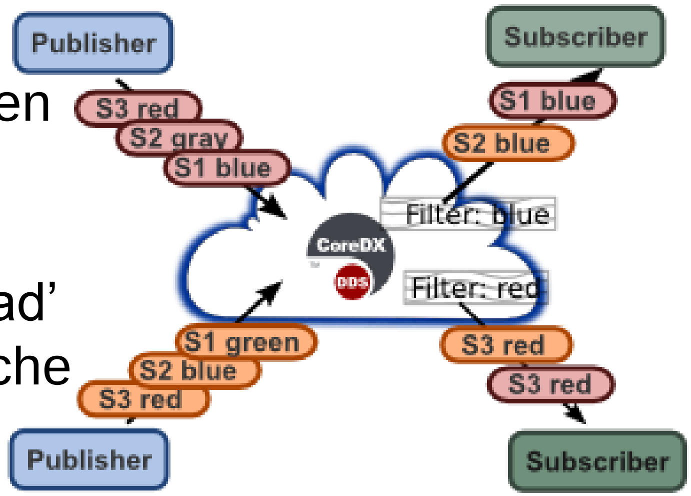
DDS Protocol:
Redundancy / Failover

- DDS provides configurable **redundancy** of published data
  - Subscribers receive one copy of data, even when multiple publishers are publishing (duplicate) data.
  - Publishers can fail, and subscribers will automatically receive data from remaining, active publishers



DDS Protocol:
Data Filtering

- Kinds of Filters:
  - Content Filters are specified by an SQL-like query
    - Same syntax as the “WHERE” clause in an SQL query
    - Filters can be applied at Reader or Writer
  - Time Based Filters are specified by a duration for minimum separation between samples
  - Read Filters allow the application to ‘read’ specific kinds of data from the Data Cache
    - Previously seen / Not seen
    - New data / Old data
    - Alive / Dead
    - Matching a content-based query



DDS Performance and Scalability

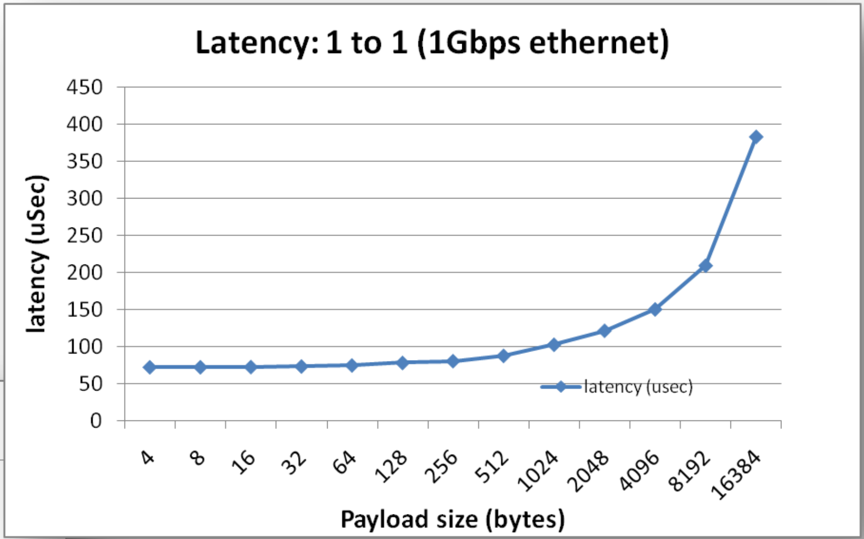
DDS Performance
Protocol Characteristics

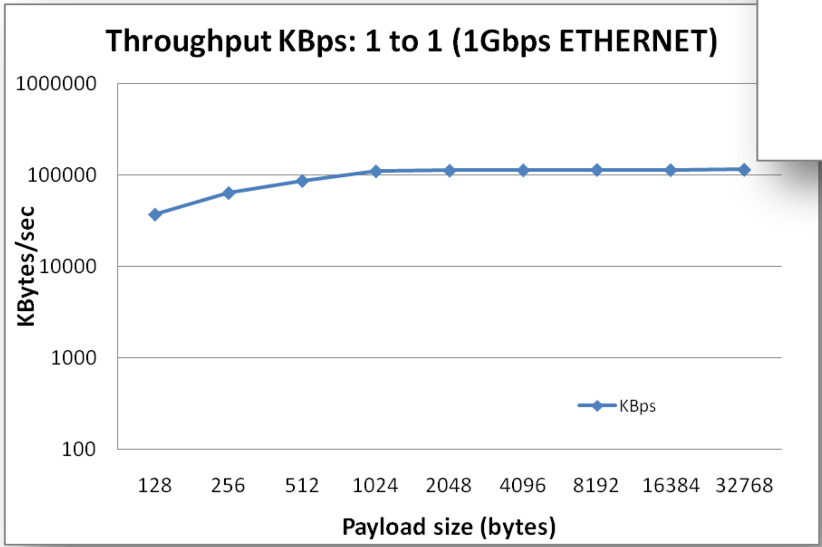
- DDS Specification written for near real-time and real-time systems
    - Specifies minimal data copies
        - Buffer loaning
    - Specifies compact encoding on the wire
        - Reducing bytes on the network
    - Specifies light-weight notification mechanisms
        - Asynchronous and Synchronous options available
    - Data types are specified at compile time
        - Allows optimized: marshalling, filtering, searching
    - Specifies UDP multicast (best effort and reliable protocols)
        - Reduced network overhead
        - High performance scalability
    - Intermediate brokers are not required
        - Allows direct peer-to-peer communications

DDS Performance: Network

Typical DDS Latencies:
50 – 200 microseconds

Typical DDS Throughput:
500 – 950+ Mbps





Typical DDS Stats:

100,000+ individual messages delivered per second – **to one subscriber!**

Multicast increases these numbers

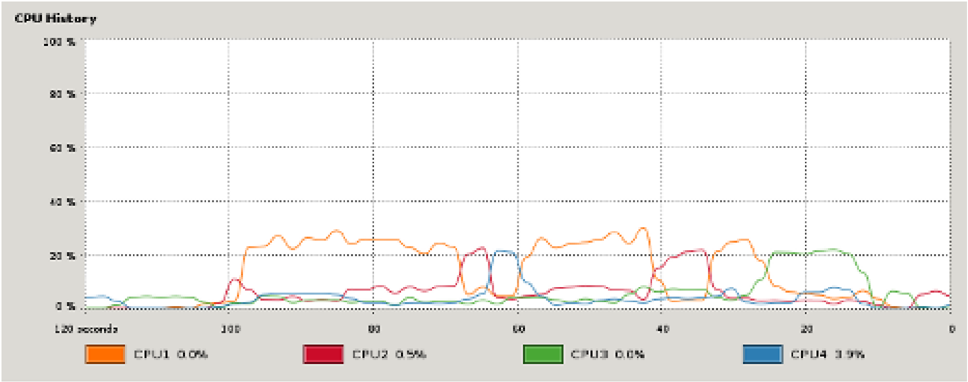
Batching increases these numbers

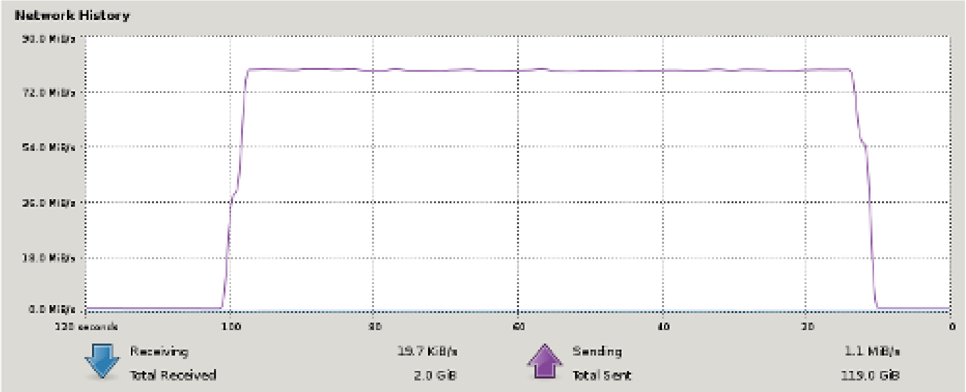
TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

DDS Performance: CPU

Long running throughput test

- TX1024 byte messages
- Total throughput of over 700 Megabits/sec
- CPU utilization of a single core
  - doesn't exceed 30%
- With 4 cores, this equates to less than 10% of total CPU





TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

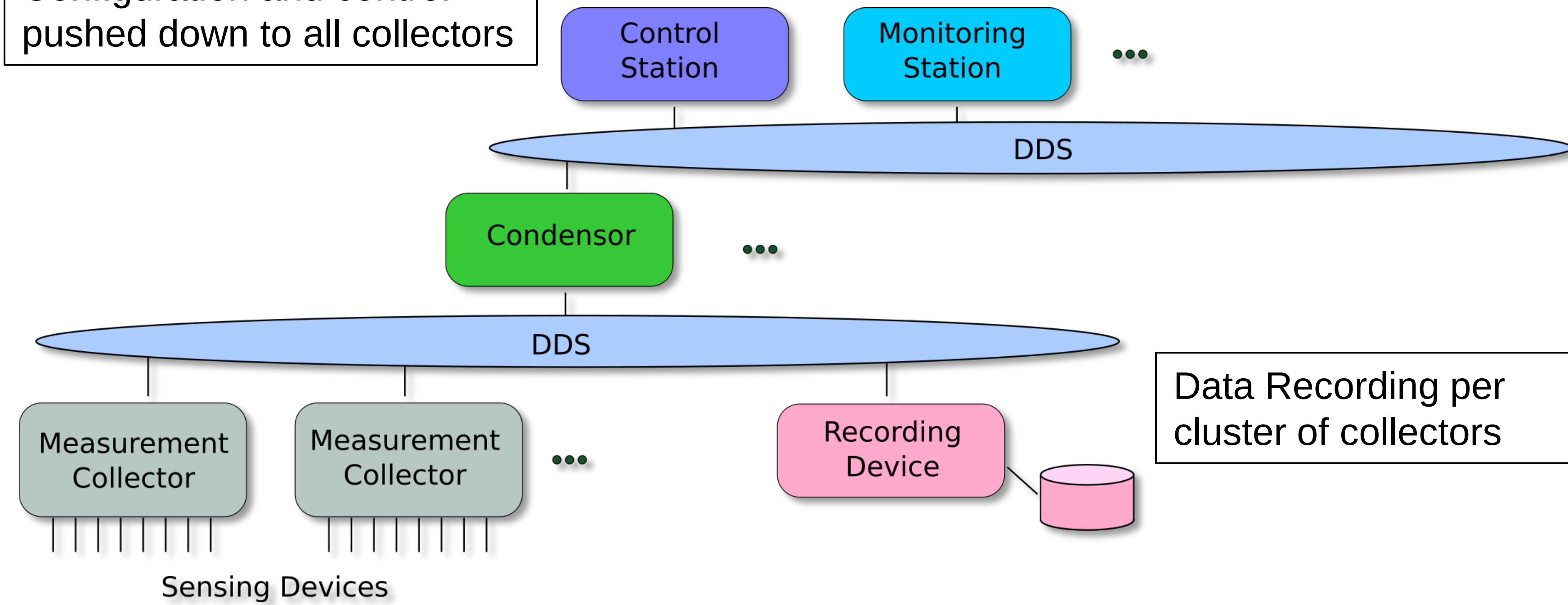
DDS Scalability Protocol Characteristics

- Scalability across all aspects
  - Network nodes, participants, topics, data instances
- Writer-side filtering
  - Only data of interest is sent on the network to the reader
- Unicast Scalability
  - Efficient UDP-based protocol with shared sessions
- Multicast Scalability (including reliability)
  - NACK-only reliability

- Additional tools to help manage large DDS networks
  - Bridging Tools: Extend DDS communications across multiple physical and logical network
  - Federation (data, discovery): Consolidate connections to reduce network, CPU, and memory usage for large numbers of DDS Entites

DDS Scalability Use Case:
Industrial Test Environment

Configuration and control pushed down to all collectors



Data Recording per cluster of collectors

15,000 total sensor measurement points
100-200 Hz update rate per measurement

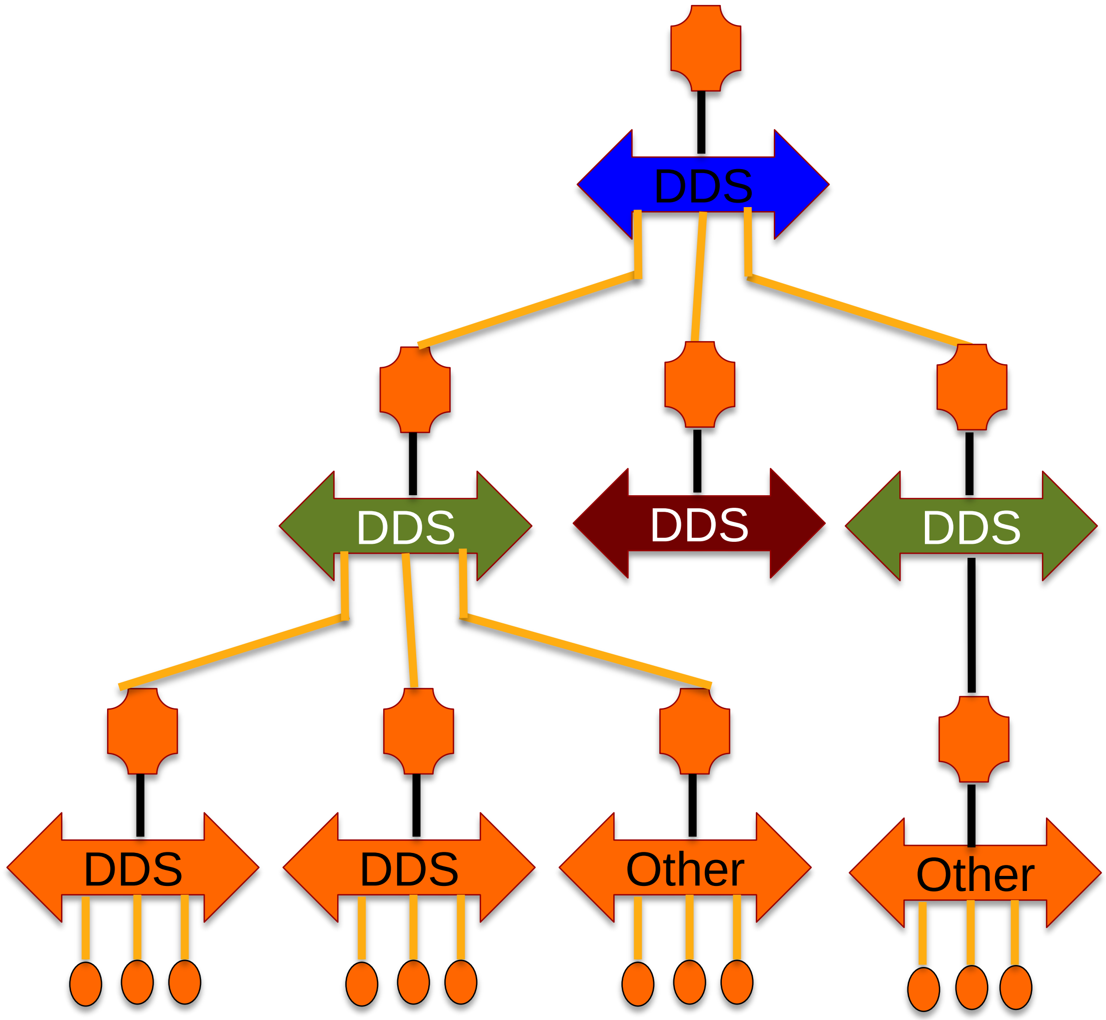
TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

Comparing
DDS and other IoT Technologies

	DDS	MQTT	AMQP	RESTful HTTP	CoAP	ZeroMQ
Designed for	Controlling & monitoring cyber-physical systems	Connecting remote devices to IT systems (telemetry)	Enterprise messaging (IT systems)	Serving web content	REST on constrained devices and networks	Socket-like high-performance messaging API
Peer-to-peer	✓			✓	✓	✓
Flexible Transport Protocols	✓				✓	✓
Publish/subscribe	✓	✓	✓		F	✓
Reliable multicast for scalability	✓					✓
Real-time QoS	✓					
Dynamic Discovery	✓					
Security built into the protocol	✓					

	DDS	MQTT	AMQP	RESTful HTTP	CoAP	ZeroMQ
[illegible] jitter	✓			✓	✓	✓
Support for latency-controlled transport protocols (such as UDP rather than TCP)	✓				✓	✓
Shared memory transport for efficient intra-node messaging	✓					
Publish/subscribe for efficient delivery of cyclic updates and low-latency delivery of asynchronous events	✓	✓	✓		F	✓
Pub/sub over multicast for low-latency broad data distribution	[illegible]					[illegible]
Reliable multicast for critical data and commands	✓					✓
Missed deadline notifications for hard real-time apps	✓					
Control over latency/throughput tradeoffs	✓					
Network and CPU efficient serialization (minimal run-time metadata)	✓					
Support for Resilient and Mission-Critical Systems						
No single point of failure or failover	✓			✓	✓	✓
Publish concurrently over redundant networks	✓					
Automatic failover between redundant publishers	✓					
Real-time notification when endpoints join or leave	✓					
Decoupled (islanded) subsystems continue to operate independently	✓			✓	✓	✓
Reconnected components/subsystems automatically synchronize state	✓					
Support for Deployment at Edge and Autonomy						
No centralized services to deploy or manage	✓			✓	✓	✓
Automatic discovery for zero configuration	✓				✓	
Communicate over unreliable networks (best effort)	✓	MQTT-SN only			✓	
Communicate reliably over unreliable networks	✓	MQTT-SN?			✓	
Support non-IP networks (e.g., radio, satellite)	✓	MQTT-SN?			✓	
Send data simultaneously over multiple transport protocols and networks (e.g., intra-machine, LAN and WAN)	✓					
Filter unsubscribed data at publisher to minimize network overhead	✓				✓	
Security						

DDS Connects
Diverse Platforms of the IIoT



DDS:

- Standardized platform support and interoperability
- Advanced protocols to support d2d, d2c, c2c communications
- Performance and Scalability to meet IIoT requirements

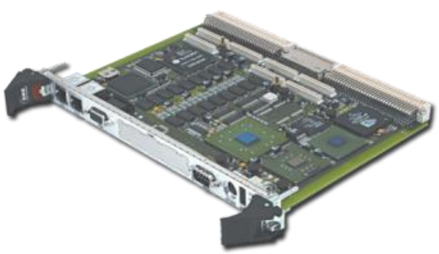
Twin Oaks Computing: Background

- Headquartered in Colorado, USA
  - World Wide Deployments
- Highly experienced executive and technical teams
- Background in embedded communication technologies
  - DDS, RTPS
  - Networking protocols
  - Device drivers
  - Embedded computing environments
- Creator of the World Leading Middleware for Safety Critical Systems: CoreDX DDS
  - Commercially Available since 2008
  - Fully Interoperable and Standards Compliant since 2009
  - 750,000+ deployments

TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

Questions?



CoreDX™ DDS