Product Advancements for Scaling and Securing IIoT Systems

Erin McManus  
PRINCIPAL ENGINEER

Fernando Crespo Sanchez  
PRODUCT ARCHITECT
Outline

Wide Area Network Connectivity
- New scalable and secure solution for Wide Area Network connectivity including 5G and other cellular networks

Health Management
- Advancements to bring better observability into Connext distributed systems

Scalability
- New features and improvements to scale to thousands of endpoints

Accessibility
- Language bindings and API advancements

COMING SOON IN 2021
Wide Area Network Connectivity

Connect from anywhere
Use Cases: Remote Device Management, Monitoring, Analytics and Assistance

Vehicle at the Edge

Cloud

INTELLIGENT LOGGING AND ANALYTICS

ANALYTICS & ML, STORE LOGGING, SERVICE DELIVERY

Secure Connext Databus

Remote Development & Operations

MULTI-VEHICLE MONITORING, ASSIST & CONTROL, UPDATES, HMI

WAN Environments Present Unique Challenges

**SCALABILITY**
The number of endpoints running in a WAN environment can be orders of magnitude bigger than the endpoints running in a closed LAN-based system.

**ROAMING AND CONNECTIVITY ACROSS DIVERSE NETWORKS**
Devices can roam between different networks, including cellular networks, changing addresses and connectivity conditions.

**SECURITY**
WAN environments are susceptible to more cyber-attacks as data traverse public networks and infrastructure.

**PERFORMANCE**
Large variations in latency and bandwidth availability make communication unreliable for performance-sensitive applications.

**SCALABILITY**
The number of endpoints running in a WAN environment can be orders of magnitude bigger than the endpoints running in a closed LAN-based system.
Challenge: Maintain Bidirectional Connectivity While Moving Across Network Boundaries

- Home Wi-Fi
- Cellular
- Public Wi-Fi

Connext Databus

## Connext DDS WAN Connectivity Solution

### Core

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underlying IP Transports</td>
<td>UDP (Connext also works with TCP)</td>
</tr>
<tr>
<td>NAT Traversal</td>
<td>Facilitates peer-to-peer connections, including external-internal and internal-internal asymmetric</td>
</tr>
<tr>
<td>Network Transitions</td>
<td>Handles changes of IP addresses while roaming</td>
</tr>
<tr>
<td>Delivery Controls</td>
<td>Many options to control communication reliability and speed (Reliable, Partially Reliable, Best Effort)</td>
</tr>
</tbody>
</table>

### Endpoint Discovery

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peer Discovery</td>
<td>Ability to discover endpoints</td>
</tr>
<tr>
<td>Signaling and Negotiation</td>
<td>Exchange of capabilities and candidate addresses</td>
</tr>
<tr>
<td>Interoperable with RTI Discovery Service (CDS)</td>
<td>Discovery at large scale, where both endpoints are behind Cone NATs; Simplified directory service that facilitates endpoint discovery</td>
</tr>
</tbody>
</table>
# Connext DDS WAN Connectivity Solution

## Performance and Scalability

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream Multiplex</td>
<td>Handles multiple streams within a connection to avoid head-of-line (HOL) blocking problems that limit performance</td>
</tr>
<tr>
<td>Flow Control</td>
<td>Manages the rate of transmission based on endpoint and network conditions to minimize data losses</td>
</tr>
<tr>
<td>Built-in Data Compression</td>
<td>Compresses user data to save bandwidth</td>
</tr>
<tr>
<td>Bonding (Post 6.1.0)</td>
<td>Combining two or more network connections to send data in an intelligent way that allows the end user to utilize the combined bandwidth in the most efficient way.</td>
</tr>
</tbody>
</table>

## Security and Privacy

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Verify the identity of the endpoints exchanging data</td>
</tr>
<tr>
<td>Access control</td>
<td>Per Connection, and fine-grained per Data Stream</td>
</tr>
<tr>
<td>Cryptography</td>
<td>Encryption and decryption, hashing, digital signatures, etc.</td>
</tr>
</tbody>
</table>
**Connext 6.1.0 WAN Connectivity Solution**

**Real-time WAN Transport**
UDP-based transport supporting NAT traversal & network roaming

**Cloud) Discovery Service**
Facilitates endpoint discovery and NAT traversal process

**WAN Connectivity Solution**
Seamlessly and reliably share data across WANs, including cellular networks, without compromising security

No need to integrate other technologies or security technologies

Single Programming Model

No API Changes
WAN Deployment Scenario: Edge to Cloud

Vehicle

Cloud

Data Center

WAN Deployment Scenario: Peer-to-Peer Edge to Edge
Any kind of NAT

1. Client ‘A’ video samples sent to RS
2. Client ‘A’ video samples received by RS
3. RS relays video samples from Client ‘A’ to other Clients
4. Clients ‘B’ and ‘C’ receive video samples from Client ‘A’ from RS
Performance: TCP WAN vs UDP WAN over Wi-Fi

Latency in usec (San Jose to Cupertino)

Throughput in Mbps (San Jose to Cupertino)
Continuous Health Management

Bringing Better **Observability** and Faster Problem **Resolution** into Connext DDS
Health Management Activities

Resolution
Solve issues and restoring the system into service with minimal impact on service availability.

Non Intrusive Health Monitoring and Alerting
Provide real-time visibility into the system behavior to assess functionality and alert about potential issues.

Non intrusive

Observability and Controllability

Diagnosis
Debug or isolate issues to root cause analysis
Observability Challenge 1: Problem

- **Health Issue:** Viewer does not receive information from Camera

- **Known Information:**
  - **Platform.** Camera platform for camera 1 and 2 is the same (hardware and software)
  - **Online.** Camera 2 system can be pinged so it is online
  - **Logs.** Logs don’t indicate any failure
  - **Inconsistent behavior.** Viewer received information in the past from both cameras
Clock Synchronization Issue:
System is configured to use destination order by SOURCE timestamp and Connext DDS has a protection mechanism in which the data published by an application is ignored if published into the future.
Observability Challenge 1: 5.3.1 Diagnosis

To determine where the data was lost, we ran Wireshark in Viewer first, and then in Routing Service.

The data was received by Routing Service but somehow was not propagated to the viewer.

Why? Not obvious. Is it lost on the middleware stack? On the socket buffers?

Finally, we realized the source timestamp of the incoming packets was 10 minutes ahead of expected.

2-3 days across multiple teams to identify root cause in a moderately complex system.
Observability Challenge 1: 6.1.0 Diagnosis

- Warning log message and metrics generated by Routing Service indicate that it dropped samples because their source timestamp was in the future.

Warning message received and visualized by Admin Console within < 1 hour to solved.

Routing Service

Camera: 1

Camera: 2

DATABUS

Viewer

< 1 hour to solved

Camera 2 data is missing

Source timestamps

11:00

11:10

New Observability Capabilities

Available in upcoming 6.1.0 release
## Logging Improvements

### Rich and configurable logging format

<table>
<thead>
<tr>
<th>Print Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>BIT_TIMESTAMP [2020-10-15 00:38:55.168092]</td>
</tr>
<tr>
<td>Thread Name</td>
<td>BIT_THREAD_ID U000000010e0ad5c0_</td>
</tr>
<tr>
<td>Activity Context</td>
<td>BIT_ACTIVITY_CONTEXT [0x01018087,0xDC43F9E2,0xF22EE887:0x80000003{E=DW,T=HelloWorld,C=HelloWorld,D=0}</td>
</tr>
<tr>
<td>File and Line</td>
<td>BIT_LOCATION_FILELINE Writer History Driver.c:731</td>
</tr>
<tr>
<td>Function Name</td>
<td>BIT_LOCATION_METHOD PRESWriterHistoryDriver_addWrite</td>
</tr>
<tr>
<td>Backtrace</td>
<td>BIT_BACKTRACE Backtrace: #4 MyType_publisher 0x000000001058538e8 WriterHistoryMemoryPlugin_addSample + 2104 #5 MyType_publisher 0x0000000010566ff78 PRESWriterHistoryDriver_addWrite + 3576 #6 MyType_publisher 0x000000001056c048c PRESPsWriter_writeInternal + 9820</td>
</tr>
</tbody>
</table>
Logging Improvements

New **activity context** associated with log messages including information to identify the **source** of the message: Topic, Type, GUID, etc.

5.3.1 (Before)

WriterHistoryMemoryPlugin_addSample: out of order
PRESWriterHistoryDriver_addWrite: !timestamp order
PRESWriter_writeInternal: !timestamp order

6.1.0  **Activity context identifies the source (Topic, Type, GUID, etc)**

[0x0101D82C, 0x034CEB6D, 0xC6A44AA8: 0x80000003{E=DW, T=HelloWorld, C=HelloWorld, D=0}| WRITE]
WriterHistoryMemoryPlugin_addSample: out of order
[0x0101D82C, 0x034CEB6D, 0xC6A44AA8: 0x80000003{E=DW, T=HelloWorld, C=HelloWorld, D=0}| WRITE]
PRESWriterHistoryDriver_addWrite: !timestamp order in topic 'HelloWorld'
[0x0101D82C, 0x034CEB6D, 0xC6A44AA8: 0x80000003{E=DW, T=HelloWorld, C=HelloWorld, D=0}| WRITE]
PRESWriter_writeInternal: !timestamp order
Logging Improvements

• Built-in backtraces help diagnose problems

• Automatically enabled for precondition or fatal errors

Backtrace:

#4  MyType_publisher  0x00000001058538e8 WriterHistoryMemoryPlugin_addSample + 2104
#5  MyType_publisher  0x000000010566ff78 PRESWriterHistoryDriver_addWrite + 3576
#6  MyType_publisher  0x00000001056c048c PRESPsWriter_writeInternal + 9820
#7  MyType_publisher  0x00000001051f5d80 DDS_DataWriter_write_w_timestamp_untyped_general + 2384
#8  MyType_publisher  0x0000000104f99bcb HelloWorldDataWriter_write_w_timestamp + 75
#9  MyType_publisher  0x0000000104f93713 publisher_main + 915
#10 MyType_publisher  0x0000000104f93903 main + 99
#11 libdyld.dylib     0x00007fff64ea83d5 start + 1
#12 ???               0x0000000000000001 0x0 + 1

[2020-10-15 00:38:55.165457] U000000010e0ad5c0_ 0x01018087,0xDC43F9E2,0xF22EE887:0x80000003{K=DW,T=HelloWorld,Y=HelloWorld,D=0}|WRITE]
Mx16:Memory.c:7433:WriterHistoryMemoryPlugin_addSample:RTI0x2161002:out of order
Monitoring Improvements: Instance Metrics

- Customer question without a clear answer in a running system:

  “How many patients are admitted, how many are discharged, is the system working as designed?

  DDS Translation

  “How many instances are registered, how many are disposed, is the system working as designed?
Monitoring Improvements: Instance Metrics

- In 6.1.0, the user will be able to answer this question by inspecting instance metrics.
- Also, available in Monitor UI.

```c
struct DDS_DataReaderCacheStatus {
    ...
    DDS_LongLong alive_instance_count;
    DDS_LongLong alive_instance_count_peak;
    DDS_LongLong no_writers_instance_count;
    DDS_LongLong no_writers_instance_count_peak;
    DDS_LongLong disposed_instance_count;
    DDS_LongLong disposed_instance_count_peak;
    DDS_LongLong detached_instance_count;
    DDS_LongLong detached_instance_count_peak;
};
```

```c
struct DDS_DataWriterCacheStatus {
    ...
    DDS_LongLong sample_count_peak;
    DDS_LongLong sample_count;
    DDS_LongLong alive_instance_count;
    DDS_LongLong alive_instance_count_peak;
    DDS_LongLong disposed_instance_count;
    DDS_LongLong disposed_instance_count_peak;
    DDS_LongLong unregistered_instance_count;
    DDS_LongLong unregistered_instance_count_peak;
};
```
Monitoring Improvements: Other Metrics

"Why am I not receiving data?"

- DataWriterProtocolStatus and DataReaderProtocolStatus extended to include metrics related to fragmented data.

- DataReaderCacheStatus and DataWriterCacheStatus extended to provide information about all samples (relevant and not relevant) dropped by Connext DDS.
Built-in Enhanced Network Capture

Wireshark is widely used at development time to capture and analyze RTPS packet traces

Limitations capturing traffic with Wireshark

- Capturing is usually restricted to users with root privileges
- Wireshark and tcpdump are often not available in production or deployed embedded targets
- Wireshark cannot capture traffic over shared memory
- Not very useful when encryption is enabled
Built-in Enhanced Network Capture

**Connext 6.1.0** release resolves capturing limitations to help with the debugging process

Capture inbound and/or outbound network traffic from multiple Domain Participants

- No need to have root access
- Available for all platforms
- Available for all transports including shared memory, TCP, UDP, etc
- Generates PCAP files that can be analyzed with Wireshark
- Security friendly: RTPS packets decryption
Built-in Enhanced Network Capture

• Programmatic API

```cpp
rti::util::network_capture::enable();
rti::util::network_capture::start(participant, "MyCapture.pcap");
rti::util::network_capture::stop(participant);
rti::util::network_capture::disable();
```
Built-in Enhanced Network Capture
Heap Monitoring

• Heap monitoring was introduced in Connext 6 to monitor memory allocations and debug unexpected memory growth

• Typical problem that can be debugged with heap monitoring:

  “Creating (and destroying) DataReaders is causing memory growth in our application. ... The growth seems to be in native space, as java heap remains consistent.”
Heap Analysis

There is a 16 byte repeated non-pool allocation of a structure with type DDS_SqlTypeSupportGlobalUnion.

“Creating (and destroying) DataReaders is causing memory growth in our application. ... The growth seems to be in native space, as java heap remains consistent.”

“Customer was creating both, DataReaders and ContentFilteredTopics, but only deleting the DataReaders"
Heap Analysis

Snapshot analysis is complicated. It usually requires sending snapshots to RTI

New Heap Analyzer
A cmd line tool for heap snapshot analysis

- Facilitates memory growth analysis
- Enables user debugging
Heap Analyzer

Process memory analysis

Potential Sources of Memory Growth Candidates grouped by topic_name
Scalability

Meet the changing needs of IIoT systems in the future
The What and Why of Scalability

Allowing *You* to build systems that continue to perform as your resource needs evolve and grow.
Bandwidth: User Data Compression

Pre-6.1.0:
• Limited Bandwidth Transport Plugin Add-on
• Compress with each send

6.1.0:
• No Additional Libraries needed
• 3 Built-in Compression Algorithms: ZLib, BZip2, LZ4
• Compress once, send n times

```c
struct DDS_CompressionSettings_t {
    DDS_CompressionIdMask compression_ids;
    DDS_UnsignedLong writer_compression_level;
    DDS_Long writer_compression_threshold;
};

struct DDS_DataRepresentationQosPolicy {
    ...
    struct DDS_CompressionSettings_t compression_settings;
};
```
Bandwidth: Compression Performance

Live Data Throughput in a 1Gbps Network (Logging data)
Bandwidth: Decoupling Reliability and Durability

Problems:
● There was previously no way to have strict-reliability for live data and a limited sample history for late-joiners
● Even if you do not require strict-reliability, there was no way to separate the reliability window from the late-joiner windows

Pre-6.1.0:
  – Both configured with a single depth parameter in HistoryQosPolicy
Problem: Replacing Unacknowledged Samples

DataWriter

History.depth

DataReader

1

1
**Problem:** Replacing Unacknowledged Samples
**Problem**: Replacing Unacknowledged Samples

DataWriter

History.depth

DataReader

The DataReader never receives 2

Late-Joining DataReader

(DataWriter's depth)
**Problem:** Too many durable samples

History.kind = KEEP_ALL

DataWriter

DataReader
**Problem**: Too many durable samples

```
History.kind = KEEP_ALL
```

DataWriter

DataWriter

DataReader

```
1  2
```

1  2
Problem: Too many durable samples

History.kind = KEEP_ALL

DataReader

1 2 3

DataWriter

1 2 3
**Problem:** Too many durable samples

- History.kind = KEEP_ALL
- DataWriter
- DataReader
- Wasted Bandwidth
- Late-Joining DataReader
- History.depth
**Bandwidth: Decoupling Reliability and Durability**

6.1.0: New DurabilityQosPolicy.writer_depth QoS

- Reliability window (KEEP_LAST) ⇒ HistoryQos.depth
- Durability window ⇒ DurabilityQos.writer_depth
**Solution:** Decouple Reliability and Durability Depths

DataWriter

History.depth

Durability.writer_dept

DataReader

1

©2017 Real-Time Innovations, Inc.
**Solution**: Decouple Reliability and Durability Depths

DataWriter

History.depth

Durability.writer_dept

DataReader

1

...
Solution: Decouple Reliability and Durability Depths

DataWriter

DataReader

Late-Joining DataReader

History.depth

Durability.writer_dept

©2017 Real-Time Innovations, Inc.
Memory: DataReader Instance Replacement Policy

Bound the resources consumed by your applications while supporting dynamic and unbounded sets of data

```cpp
enum DataReaderInstanceRemovalKind {
    NO_INSTANCE_REMOVAL,
    EMPTY_INSTANCE_REMOVAL,
    FULLY_PROCESSED_INSTANCE_REMOVAL,
    ANY_INSTANCE_REMOVAL,
};

struct DataReaderResourceLimitsInstanceReplacementSettings {
    DataReaderInstanceRemovalKind alive_instance_replacement;
    DataReaderInstanceRemovalKind no_writers_instance_replacement;
    DataReaderInstanceRemovalKind disposed_instance_replacement;
};
```
Accessibility

Access Connext from a variety of development environments and languages
New and Improved Language Bindings

- Fully redesign, modern, multi-platform .NET API
- New experimental Python API
- New RPC for IDL interfaces. C++11 improvements
New .NET API

New fully redesigned, OMG-standard, .NET 5 compatible C# API

Compatibility
• Built for .NET Standard 2.0 & compatible with .NET 5, .NET Core, Unity, .NET Framework
• Runs on Linux, macOS, Windows…
• Deployed with Nuget
• OMG-standard IDL mapping

API
• Fully redesigned with .NET best-practices, naming conventions, & idioms.
• Generics
• Entities are IDisposable
• Standard collection interfaces
• Status updates and Conditions use events
• Immutable types
• OMG standardization in progress

using DomainParticipant participant = DomainParticipantFactory.Instance.CreateParticipant(domainId);

var provider = new QosProvider("hello_world.xml");
Topic<TopicData> topic = participant.CreateTopic("Example HelloWorld", provider.GetType("HelloWorld"));

Subscriber subscriber = participant.CreateSubscriber();
DataReader<TopicData> reader = subscriber.CreateDataReader(topic);

StatusCondition statusCondition = reader.StatusCondition;
statusCondition.EnabledStatuses = StatusMask.DataAvailable;
statusCondition.Triggered += condition =>
{
    using var samples = reader.Take();
    foreach (var sample in samples)
    {
        Console.WriteLine("Received: {sample}"postcode);
    }
}

var waitset = new WaitSet();
waitset.AttachCondition(statusCondition);
waitset.Dispatch(Duration.FromSeconds(4));

Preview available @ https://www.nuget.org/packages/Rti.ConnexDds/
New Python API (experimental)

Full access to Connext DDS from Python

Python-friendly design built on the Modern C++ API

Full access to Connext DDS features:
- Dynamic Data
- Built-in types
- DDS Entities in code and XML
- DDS QoS in code and XML
- Content Filters
- Built-in discovery Topics
- Status updates
- Listeners
- Wait Sets
- Conditions

Available now @ https://github.com/rticommunity/connextdds-py
Read about it @ https://www.rti.com/blog/introducing-the-rti-python-api
New RPC for IDL interfaces (experimental)

CarControl.idl

```idl
struct DiagnosticOptions {
    // ...
};

@service
interface CarControl {
    attribute int32 speed;
    boolean run_diagnostics(in DiagnosticOptions options);
};
```

CarControl_service.cxx
(generated by rtiddsgen)

```cpp
class MyCarControl : public CarControl {
public:
    void set_speed(int32_t speed) override { speed_ = speed; }
    int32_t get_speed() override { return speed_; }

    bool run_diagnostics(const DiagnosticOptions &options) override {
        return check_car_status(options);
    }

private:
    int32_t speed_;
};
```

CarControl_client.cxx
(generated by rtiddsgen)

```cpp
CarControlClient client;
client.wait_for_service();

client.set_speed(client.get_speed() * 2);
if (!client.run_diagnostics(options)) {
    cout << "Attention! Car diagnostics failed!\n";
}
```

Coming soon for C++11
Connector improvements (py, js)

• Instance management
  – Instances can now be disposed and unregistered
  – Subscriptions can look up disposed keys

• Dynamic Library Support for pluggable RTI components
  – Monitoring
  – Security

• JavaScript: Node 12 support
  – Support for Node 14 coming later
Try a full version of Connext DDS for 30 days

TRY CONNEXT AT RTI.COM/DOWNLOADS

Includes resources to get you up and running fast
Stay Connected

rti.com
Free trial of Connext DDS

@rti_software

rtisoftware

@rti_software

connextpodcast

rti.com/blog