WHITEPAPER

IIoT Protocol: Comparing DDS and MQTT

OVERVIEW

Modern organizations that provide any automated functionality grapple with two worlds:

- **Operational Technology (OT)** – the traditional physical world composed of machines, electromechanical devices, manufacturing systems and other industrial equipment.
- **Informational Technology (IT)** – the digital information world that uses servers, storage, networking and other devices to run applications and process data.

Now, software-defined systems are blending these two worlds, requiring seamless and secure data flow between IT and OT systems. This integration requires a standardized approach to communication to allow for interoperable interfaces, data translation layers, and secure communication channels. The data framework that connects the systems acts as the digital glue that enables open data exchange and real-time decision-making between enterprise IT systems and industrial processes.

The evolution of communication models has been marked by a gradual reduction in coupling between communicating entities, which are moving from legacy point-to-point communication to more abstract, data-centric models. Each new model has addressed the weaknesses of its predecessor by introducing improved scalability, reliability, and flexibility in communication. Two popular communication protocols used today are MQTT (Message Queuing Telemetry Transport) and DDS™ (Data Distribution Service). This paper will explore the similarities and differences between MQTT and DDS in terms of architecture, performance, scalability, and suitability for various applications.

The most impactful way to compare these two technologies is to understand what DDS and MQTT were built for in the first place.

MQTT has its roots embedded in one of the words within its name: Telemetry. The protocol and its corresponding implementations were created to connect smart devices to the cloud, where elastic computing can be used to collect and analyze data from the field. Its hierarchical topic structure lends itself to the tracking of many devices out in the field and leverages standard internet-based protocols for sending and receiving data.

DDS was built specifically for the localized real-time distribution of data for applications requiring more real-time control. DDS provides all of the functionality of MQTT and much more. DDS can be used for telemetry and for device tracking, but it is also used for more advanced applications such as robotic surgery and real-time automotive vehicle control. MQTT was never built for those types of applications and fails in any attempt to provide such functionality.

Let's now look at the specific architecture for each protocol.

ARCHITECTURE

Architecture is the backbone upon which good software stands, defining its structure, organization, functionality, and reliability. A robust architecture ensures efficient communication among different components, promotes code reusability, and enforces best practices, leading to reduced development time and higher quality code. Moreover, a clear architecture enhances collaboration among developers, allowing them to work in tandem and contribute effectively to the project. In essence, architecture provides the blueprint for the software's longevity, performance, and overall success, making it a cornerstone of any software-led development endeavor.

The technology choices significantly influence the type of architecture that can be constructed within any system or application. Different technologies offer unique capabilities, constraints and performance characteristics, which in turn dictate the architectural decisions. Thus, technology choices serve as the foundation upon which engineers implement the architecture, shaping everything from data storage and processing methods to communication models and deployment strategies. Making informed technology choices is crucial to aligning the architecture with the intended goals and requirements of the system.

Future-proof architectures for any kind of automation must be distributed, scalable, fault tolerant, deterministic, performant, and enable zero-trust security. Now, architects are also faced with a new requirement: the system must be architected for constant change. DDS and MQTT provide two distinct approaches to distribute data within a system: message-centric and data-centric.

- **MQTT** follows a lightweight message-centric, broker-based, publish-subscribe model. It uses a broker-based approach where publishers send messages to a broker, which then distributes the messages to relevant subscribers based on user-defined topics. MQTT relies on the broker for discovery and dynamic participant join/leave (e.g., using additional technologies such as DNS or manual configuration). MQTT focuses on the transmission of opaque data payloads with limited delivery tunability, making it suitable for applications on resource-constrained devices and unreliable networks where messages are published and consumed independently. However, the use of opaque data payloads makes it difficult to upgrade and modify the application for demanding, changing needs.

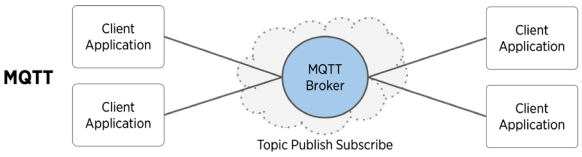

Figure 1: MQTT Communication Model

- **DDS** employs a data-centric publish-subscribe model. Data-centric means the primary interface between applications is the data itself, not the messages. This allows applications to easily join the overall distributed system as they already know what data is available for publishing and subscribing. DDS also uses a distributed approach in which participants communicate directly with each other, eliminating the need for a centralized broker. This approach completely eliminates single points of system failure. DDS also includes built-in mechanisms for discovery, allowing new publishers and subscribers to dynamically join and leave the system while maintaining awareness of changes in the network. The DDS publish-subscribe model provides fine-grained control over data communication with Quality of Service (QoS) and offers more flexibility in data distribution, discovery and data security.

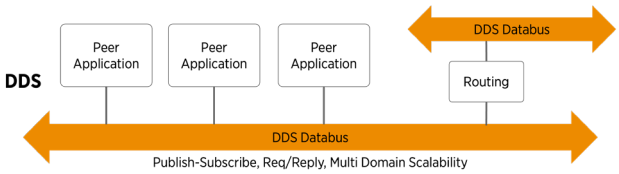

Figure 2: DDS Communication Model

The following sections compare DDS and MQTT with core system objectives:

TOPICS

DDS and MQTT both use topics to define the primary communication paths for applications to be able to talk with each other. They play a significant role in facilitating communication and coordination among distributed components. Topics act as communication channels, allowing producers to publish data and subscribers to express interest in specific types of data, promoting loose coupling between components.

In DDS, a Topic is a key concept. A DDS Topic represents a specific type of data that is being exchanged within a DDS system. Topics define the format or structure of the data and are used to categorize and organize the data being sent and received within a DDS system, allowing different parts of the system to subscribe to or publish the data of interest.

Each topic in a DDS system is associated with a specific data type, which defines the structure and content of the data. Topics enable publishers to send data to specific subscribers who have expressed an interest in that particular Topic, facilitating efficient and selective data communication in real-time and distributed systems. As the system evolves, so can the data definitions. This is supported by a Type Extensibility mechanism that provides both backward and forward compatibility of data types.

In MQTT, a Topic serves as a logical channel or subject to which messages are published and from which messages are subscribed. Topics are represented as hierarchical strings, typically organized in a tree-like structure, using a slash ("/") as a delimiter. These Topic hierarchies help categorize and organize the messages exchanged within an MQTT-based system. Unlike DDS, MQTT is not aware of the message content. MQTT Topics only help with information organization.

Both MQTT and DDS use Topics to categorize and organize data. In both instances, Topics play a critical role by facilitating the selective dissemination of data to interested parties, ensuring efficient, scalable and targeted communication. While both use Topics as a means of organizing and routing data, they have significant differences in their implementations and intended use cases.

DDS AND MQTT DIFFERENCES

Keys

- **DDS** introduces the concept of DDS Keys, which are used to uniquely identify data instances within a Topic. Each data sample can be associated with a key, and subscribers can express interest in receiving updates for specific instances identified by their keys. In addition, DDS provides knowledge to subscribing applications for when instances are created and deleted, thus enabling a higher level of interaction between publishers and subscribers of data. DDS Keys are useful in scenarios where there is a need for fine-grained control over the specific data instances of which a subscriber is interested.

- **MQTT** does not have an equivalent concept of Keys for identifying specific data instances. Subscribers receive all messages published to the Topics to which they are subscribed, without the ability to filter based on specific data instance identifiers. For MQTT, filtering is available but only done at the Topic level.

Content Filtering

- **DDS** supports content filtering, allowing subscribers to express interest in specific data based on content conditions. This enables subscribers to receive only the data that matches certain criteria, providing a high level of flexibility in data selection. Content filtering in DDS is achieved through the use of SQL-like expressions that allow subscribers to specify conditions that data must meet to be of interest. Learn more from the RTI Community Knowledge Base.

- **MQTT** supports filtering based on Topic hierarchy using wildcards (+ and #), but it doesn't provide the same level of content filtering capabilities as DDS. Subscribers receive all messages published to a subscribed Topic, regardless of their content.

While MQTT is not inherently designed for deterministic communication, it can still be used in control applications under certain conditions and with appropriate design considerations. See the Appendix for additional analysis of Real Time Control with MQTT.

PERFORMANCE

Performance is crucial in distributed computing because it influences the effectiveness, efficiency, and reliability of complex systems. A well-performing distributed system can handle larger workloads, provide a seamless user experience, optimize resource usage, and contribute to achieving business goals.

The communication protocol should enhance system performance by ensuring efficient data transmission, minimal latency and effective load balancing. Conversely, a poorly designed or inefficient framework can introduce bottlenecks, delays, and increased network traffic, which will negatively affect overall system performance. Both protocols are designed for distributed communication, making them suitable for applications where data needs to be transmitted between devices or components in a network, but their differences are important to understand.

- **DDS** is designed to handle high-performance (high data throughput along with low latency) and real-time data exchange. Because of the use of direct peer-to-peer connections between publishers and subscribers of data, there are no separate services or brokers running in the middle to affect latency, reliable data delivery, and data security. Additionally, DDS supports efficient data filtering (including content filtering) to reduce unnecessary data on the wire, enable multicast communication to support one to many communication, and provide reliable delivery mechanisms to ensure data delivery for critical components. DDS can handle large data sets and provides fine-grained control over data Quality of Service (QoS) parameters, making it suitable for applications with stringent performance requirements. DDS allows more choices in QoS, enabling flexibility in the control choices made to meet those performance requirements. DDS provides over 23 high-level QoS policies to meet the differing requirements of each communication channel. These policies include options such as ownership, durability, data caching, and resource management, suitable for applications requiring complex data flow delivery and storage.

- **MQTT** is optimized for low bandwidth network environments. It uses a simple header structure, resulting in smaller message sizes and reduced network overhead. However, it may not be ideal for applications with high data throughput or strict latency requirements. Both throughput and latency are impacted by using a broker architecture that performs worse with increasing data requirements. MQTT allows developers to configure one QoS policy that determines the reliability of the messages.

SCALABILITY

Scalability is vital in distributed systems, as it enables the system to grow and adapt in response to changing demands, user expectations, and technological advancements. A scalable system ensures that performance is maintained, resources are optimized, and user experiences are consistently positive for long-term success and sustainability.

The communication protocol significantly influences the scalability of a system. It facilitates the addition of new resources, nodes, or components without causing excessive overhead or bottlenecks. Conversely, if the communication structure is inadequate, it can hinder scalability by introducing latency, contention, and/or inefficient data routing, thereby limiting the system's ability to handle increased loads or growing user demands.

- **DDS** is inherently scalable, as it operates in a distributed manner. It can support a large number of participants, and its peer-to-peer communication model allows for efficient data distribution in distributed systems. Through the use of Domains and Domain Partitions, DDS can scale to accommodate complex and dynamic system environments. DDS has been proven to work in very high scale applications that are tracking well over 200,000 different data points in the system.

- **MQTT** is a scalable protocol that can handle a large number of publishers and subscribers. However, scalability may be limited by the capacity of the MQTT broker, which acts as a centralized point of communication. This scalability limitation is directly affected by the level of activity of data through the broker. The more data sent through the broker, the slower it is at delivering data to consumers. This translates into non-deterministic latency for the delivery of data, limiting it to applications that do not need real-time control.

RELIABILITY

Reliable data delivery is essential in distributed systems to ensure data integrity, system availability, user satisfaction, business continuity and overall system operation. By incorporating data reliability mechanisms and practices, organizations can build robust distributed systems that deliver consistent performance, withstand failures, and provide a foundation for successful and dependable operations.

To support reliable data delivery, the middleware must ensure that messages are delivered accurately, in order and consistently, even in the presence of network issues or component failures. Conversely, an unreliable communication framework can introduce vulnerabilities and disrupt data flow, leading to system instability and potential data loss. From the beginning and as the system evolves, the communication protocol must be able to consistently perform its functions and maintain data integrity, even under adverse conditions.

- **DDS** can significantly enhance the reliability of a system due to its robust communication features. With no reliance on the underlying transport to provide reliable communication, DDS is designed to ensure the dependable and deterministic data exchange among distributed components, making it well suited for applications where reliability is paramount. It provides advanced QoS settings, including options for reliable message delivery, data persistence and fault tolerance, all of which contribute to system reliability.

DDS also offers redundancy and failover mechanisms, allowing for seamless data flow even in the event of network disruptions or component failures. With its built-in reliability features and fine-grained control over communication behavior, DDS is used in mission-critical applications in various domains, including aerospace, defense, healthcare, automotive, and industrial automation.

- **MQTT** can significantly impact the reliability of a system, although it requires careful configuration to ensure robust performance. MQTT offers QoS levels that allow users to select the appropriate trade-off between reliability and overhead. While MQTT's QoS 0 (*at most once*) mode provides low assurance of message delivery, higher QoS levels,1 (*at least once*) and 2 (*exactly once*), introduce mechanisms for guaranteed message delivery, which can enhance system reliability. However, MQTT's reliability can be affected by factors such as network stability and broker availability (single point of failure). Careful consideration of these factors and appropriate QoS settings can make MQTT a reliable choice for many applications, including those in the IoT and IIoT domains.

STANDARDS AND INTEROPERABILITY

Standards and interoperability are vital in a communication framework, as they form the foundation for seamless and efficient data exchange between heterogeneous systems and devices.

Interoperability allows for the integration of legacy systems, as well as integration across diverse hardware and software platforms. This reduces the risk of vendor lock-in and promotes competition and innovation.

- **DDS** is an active, open, and well-maintained standard maintained by the Object Management Group® (OMG®). It has a strong focus on interoperability and encompasses a comprehensive body of standards designed to facilitate efficient and reliable data communication in distributed systems. The specifications within the DDS standard cover various aspects of the communication model, including the specification of data-centric topics, publish/subscribe mechanisms, on-the-wire protocol for standardized message exchange and QoS policies. The DDS standard provides a well-defined API, ensuring code portability. The DDS standard also has a well-defined wire protocol, RTPS, which ensures interoperability between DDS versions. Extensive DDS security features include fine-grained access control and data-centric security policies, suitable for applications with stringent security requirements. (*See the DDS-Security™ specification here*). The DDS standard's rich set of specifications reflects its adaptability to complex and dynamic distributed environments, making it a versatile choice for developers and industries seeking reliable and real-time data communication solutions.

- **MQTT** is overseen by various organizations including OASIS® and ISO, and constitutes a set of lightweight and efficient protocols for publish-subscribe messaging in IIoT and other resource-constrained environments. The MQTT standard defines the structure and behavior of the open protocol, specifying how clients communicate in a broker-based architecture. The standard specifies security as non-normative (*see MQTT Version 5.0 specification here*). The specification provides security features such as authentication, authorization and encryption, but security implementation may vary across MQTT brokers and devices.

USE CASES

DDS and MQTT are both used in IIoT applications that focus on the digitization and automation of industrial processes. However, due to their specific characteristics, each excels in different aspects of these use cases. The following section provides a breakdown of the strengths and weaknesses of each protocol:

DDS Use Cases

- **Real-Time Control Systems** – DDS is designed for applications where real-time control and deterministic communication are paramount. Specifically, it excels in situations where precise timing and low latency are essential, such as robotic automation and process control.

- **Complex Data Modeling** – DDS is ideal for IIoT applications with complex data structures and models. Its data-centric approach allows for the efficient sharing of structured data across distributed systems, making it well-suited for applications with intricate data requirements.

- **Highly-Reliable Systems** – DDS is often chosen for mission critical systems where reliability and fault tolerance are critical. It offers advanced QoS controls to ensure data integrity and reliability.

- **Large-Scale and Scalable Systems** – DDS can handle large-scale deployments with numerous participants and data sources, while maintaining determinism and reliability. It scales effectively for system expansion.

- **Advanced QoS** – DDS is the only standard built from the ground up to provide advanced QoS controls, allowing users to specify delivery guarantees, message persistence, and resource management with fine granularity. This is beneficial for applications with diverse and stringent QoS requirements.

MQTT Use Cases

- **Data Aggregation and Reporting** – MQTT is well-suited for scenarios that require data collection from a variety of sensors, machines or devices to be transmitted to a central system for analysis and reporting. It can efficiently gather data from distributed sources and relay it to a data hub.

- **IIoT and Sensor Networks** – MQTT's lightweight and simple publish-subscribe model make it an excellent choice for connecting a large number of devices and sensors. It is particularly effective in scenarios where there is a high device density, such as monitoring and controlling sensors on factory floors.

- **Remote Monitoring and Control** – MQTT is used for real-time monitoring and control of industrial equipment and processes. It can facilitate remote access and control, enabling operators to manage machines and processes from a central location.

- **Alerting and Notifications** – MQTT is suitable for use cases that require real-time alerts and notifications. It can quickly deliver alerts and updates to relevant personnel or systems when specific events or conditions occur.

- **Integration with Cloud Services** – MQTT integrates seamlessly with cloud platforms, making it suitable for applications that involve cloud-based analytics, data storage and processing. It is often used for sending data to cloud-based applications for advanced analytics.

The choice between DDS and MQTT depends on specific use case requirements. Architects need to consider factors such as latency, reliability, scalability, data complexity, and the need for real-time control when deciding which protocol is better suited for the application.

- **DDS** excels in applications that demand real-time control, deterministic communication, complex data modeling, reliability, or advanced QoS control. It is often chosen when precision, fault tolerant and/or advanced QoS management are critical, providing low-latency communications for resource-constrained systems to large-scale, complex systems.

- **MQTT** is well suited for use cases that involve data aggregation, IoT sensor networks, remote monitoring, cloud integration and scenarios where lightweight communication and simple integration are key priorities. It is a good choice when scalability and broad device connectivity are essential.

CONCLUSION

Communication patterns and the network architecture are critical in designing high-performance systems. The fusion of IT and OT holds immense potential for informed decision-making, automation and increased productivity. However, it also presents challenges, such as security concerns and architectural complexities.

MQTT and DDS play a pivotal role in shaping the architecture, performance, scalability, reliability, and overall success of distributed systems. DDS places a strong emphasis on real-time control, interoperability, and standardized APIs, ensuring compatibility between vendor implementations in large and distributed mission-critical systems. MQTT, with its widespread adoption, offers a wide array of client libraries and broker implementations, fostering interoperability across various platforms.

It's important to note that while MQTT can be used for non-deterministic control, there are situations where strict determinism and real-time guarantees are critical. In such cases, other communication protocols such as DDS or specialized real-time protocols, are more suitable, as they provide finer-grained control over message timing and delivery guarantees.

In the ever-evolving IIoT landscape, organizations must carefully consider their specific use case requirements when choosing between MQTT and DDS, weighing factors such as latency, reliability, scalability, data complexity, and the need for real-time control. By aligning their technology choices with their objectives, organizations can harness the full potential of IT/OT convergence and drive innovation, efficiency, and competitiveness in their operations. Ultimately, the choice between these two communication frameworks plays a crucial role in shaping the future of industrial automation and digital transformation.

To learn more about designing high-performance systems with DDS and the RTI Connext® performance advantage, visit our Connext Product Suite and Performance web pages.

APPENDIX
Terminology and Concepts

Communication Models

Communication lies at the core of information dissemination, enabling the exchange of data between entities in various domains, from computer networks to telecommunication systems. As a result, communication models have undergone a remarkable evolution, adapting to the changing demands of information exchange by creating more efficient, scalable and adaptable systems. Each new model has progressively reduced coupling and improved scalability, addressing the limitations of its predecessor.

Traditional point-to-point communication involves a direct connection between a sender and receiver, limiting the scalability and flexibility of communication systems. In contrast, the data-centric publish-subscribe model, with its dynamic subscriptions and efficient routing, represents a significant milestone in communication evolution, enabling modern applications to thrive in an ever-changing information landscape. Here is the progression from the traditional point-to-point model to the sophisticated data-centric publish-subscribe model:

Point-to-Point (Peer-to-Peer)

Individual devices or nodes can directly exchange data or resources without the need for a centralized server or intermediary.

- **Strengths** – Increased resilience because it does not rely on a single central server, making it robust in decentralized networks.
- **Limitations** – Scalability may be limited by the capacity of the MQTT broker, increased complexity as the number of connections grew, and high coupling between sender and receiver. DDS does not have a scalability limitation.
- **Coupling** – High, as each sender needed to know the exact location and characteristics of the receiver.

Client-Server

The client/server model introduced a more structured approach, with clients making requests to servers for data or services.

- **Strengths** – Provides centralized control, making it easier to manage data, security, and access permission.
- **Limitations** – Scalability limitations, single points of failure, and the need for clients to know the location and details of specific servers.
- **Coupling Level** – Moderately high, as clients were tightly coupled to specific servers.

Publish-Subscribe

Publish-subscribe introduced a level of decoupling by allowing publishers to broadcast messages to a set of subscribers who express interest in specific topics or events.

- **Strengths** – Decoupled data distribution, allowing multiple subscribers to receive updates from publishers without direct point-to-point connections.
- **Limitations** – Limited support for complex data relationships, potential message loss, and scalability challenges in large-scale systems. This still requires a central broker or mediator for message routing, potentially becoming a single point of failure.
- **Coupling Level** – Publish-subscribe reduces coupling further by allowing subscribers to be unaware of the exact location and identity of publishers. Subscribers only needed to express interest in specific topics and knowledge of brokers used for routing.

Data-Centric Publish-Subscribe

Data-centric publish-subscribe represents a modern evolution of communication models, emphasizing data-centricity and Quality of Service (QoS).

- **Strengths** – Supports complex data relationships, real-time communication, and fine-grained control over message delivery and reliability.
- **Limitations** – Complexity in configuration and potential over-engineering for simple use cases.
- **Coupling Level** – Very low, as data-centric publish-subscribe abstracts away most of the communication details, allowing publishers and subscribers applications to discover each other automatically and focusing solely on the data they need.

Data-Centric Architecture

Data centricity makes the data the means of interaction. A data-centric infrastructure must define the data it manages. Then, the infrastructure imposes rules that determine how data is structured, when data is changed, and how it is delivered. There is a notion of a "global data space," where data values, in known structures, are exchanged. The infrastructure is aware of the contents. With a data-centric approach, developers write applications that read and update entries in a logical global data space. This ensures that applications have the data where and when they need it.

Message-Centric Architectures

These architectures consider the message itself to be the means of interaction. The infrastructure is unaware of the message contents. With a message-centric approach, developers write applications that send messages between participants. Typically, these architectures create hierarchical solutions which have limited scalability, are expensive to integrate and maintain, require manual system integration, are inflexible, and typically cannot expand the changing demands of the market.

Network Architecture

IIoT (also known as Industrie 4.0) emphasizes the integration of digital technologies into manufacturing and industrial processes. It necessitates a robust network infrastructure to support real-time data exchange, autonomous decision-making, and seamless connectivity across devices and systems. Centralized, decentralized, and distributed network architectures offer distinct approaches to meeting these requirements.

Centralized Networks

Centralized networks typically rely on a single control center or data repository. They excel in providing a unified view of operations and are relatively straightforward to manage. However, they may struggle to scale effectively, face single points of failure, and can be less adaptable to rapid changes, which are critical aspects of IIoT systems.

Decentralized Networks

Decentralized networks distribute decision-making authority across multiple nodes or components, reducing the risk of single points of failure and enhancing fault

tolerance. These networks are well-suited for scenarios where autonomy and local decision-making are essential, such as in IoT ecosystems. However, they require careful coordination and synchronization mechanisms and may still face scalability limitations in large-scale implementations.

Distributed Networks

Distributed networks disperse control and data across a network of interconnected devices and systems. They offer high scalability, fault tolerance, and load distribution. These networks are an excellent fit for IIoT requirements, as they facilitate real-time data exchange, support dynamic resource allocation, and are highly adaptable to changing conditions. However, they demand sophisticated communication protocols, security measures, and data management strategies.

Reliability and Resilience

IIoT operations demand high reliability and resilience. Centralized networks may introduce single points of failure, whereas decentralized networks improve fault tolerance. Distributed networks, by their nature, offer inherent redundancy and fault tolerance, making them highly reliable in IIoT environments.

Scalability

Scalability is crucial in modern industrial settings, where the number of connected devices and data volume can grow rapidly. Centralized networks may struggle to scale efficiently, while decentralized networks offer improved scalability compared to centralized ones. Distributed networks, designed with scalability in mind, excel in accommodating large-scale implementations.

ABOUT RTI

Real-Time Innovations (RTI) is the infrastructure software company for smart-world systems. Across industries, RTI Connext® is the leading software framework for intelligent distributed systems. RTI runs a smarter world.

RTI is the market leader in products compliant with the Data Distribution Service (DDS™) standard. RTI is privately held and headquartered in Silicon Valley with regional offices in Colorado, Spain, and Singapore.

RTI, Real-Time Innovations and the phrases “RTI Runs a Smarter World” and “Your systems. Working as one,” are registered trademarks or trademarks of Real-Time Innovations, Inc. All other trademarks used in this document are the property of their respective owners. ©2024 RTI. All rights reserved. 50057 V2 0424

Your systems. Working as one.

CORPORATE HEADQUARTERS

232 E. Java Drive, Sunnyvale, CA 94089
Telephone: +1 (408) 990-7400
info@rti.com

rti.com
company/rti
rti_software
rti.com/blog
rtisoftware
connextpodcast