DDS Foundation Use Case

DDS-Foundation.org

Healthcare

DDS in Patient Monitoring

Author: Matt Grubis, Chief Engineer

GE Healthcare Patient Monitoring

March 2020

Document No. DDSF001.0320

Table of Contents

I. Executive Summary

The Data Distribution Standard (DDS) for real-time systems, published by the Object Management Group (OMG), most closely matches the stringent and challenging requirements for data connectivity of a hospital patient monitoring system. A patient monitoring system typically consists hundreds, sometimes thousands of interconnected physiological acquisition devices (bedside monitors), real-time clinical viewers with command and control of the ecosystem (central stations), and event annunciation (sounding clinical and system alarms) to a plurality of locations, based on caregiver workflows. The DDS middleware's data-centric, network-based state management is fundamental to the reliability, robustness and distributed operation of a system designed to make life-critical decisions and provide clinical information for further care.

II. The Application

Patient monitoring is more than a device connected to a patient that acquires physiological information and processes that information generating alarms. A Patient Monitoring **system** includes many, often hundreds and sometimes thousands of devices connected to patients geographically dispersed across a hospital campus. Data and processed information from each of these devices are communicated across the hospital ecosystem and are delivered to a variety of data sinks. For example, patient data flows to clinical viewing applications at desktop stations, in centralized monitoring "war rooms", on handheld devices, data repositories, clinical algorithms that further process and analyze patient data, and the Electronic Medical Record (EMR).

Illustration of a patent in the Intensive Care Unit with physiological monitors and ventilator

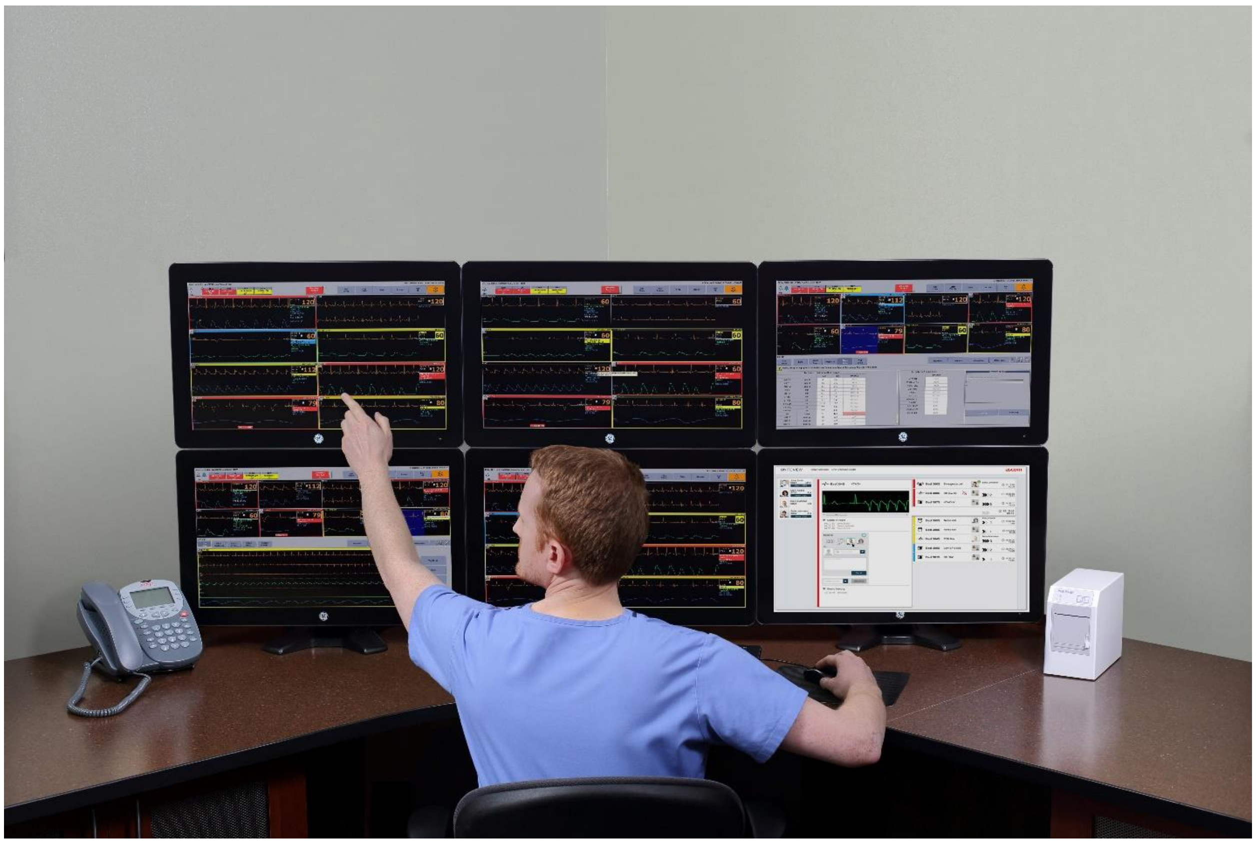
This complex and connected ecosystem is extremely dynamic in nature. Patients are continuously admitted and discharged. Devices are shutdown and restarted. Based on disease state and current conditions of the patient the physiological parameters being monitored on a patient may be increased or decreased. The algorithms are tuned to the patient and alarming-triggers modified. Devices and patients are mobile and may "wander" away from network coverage and then reenter sometime later.

The dynamic behavior mentioned above is expected when the system is operating under *normal* conditions. To bring further complexity to the situation, network switches fail, optical fiber is cut, servers lose power or fail, patient devices are abused and broken, WiFi coverage is poor and software requires lifecycle management. Above all, **a system of this nature can never be "turned off".** Patients must always be continuously monitored for life-critical events and a caregiver notified if such an event is occurring. Regardless of the dynamic behavior, the system must remain safe and effective.

Controlling the system, such as admitting and discharging patients, changing device and algorithm configurations, or managing clinical events and alarms, can be performed from almost anywhere in the ecosystem. It is possible that multiple clinicians are modifying or responding to the same patient simultaneously from different locations in the ecosystem, particularly from devices we call Central Stations.

We believe this ecosystem is the quintessential MIoT (Medical Internet of Things) paradigm.

Illustration of a clinical technician in a central monitoring room providing surveillance of 40 different patients, including the ability to manage clinical alarms and escalate if necessary.



III. Requirements of the Communication System

Considering the numerous clinical workflows and applications, the sheer scale of the system and demand of high availability, the requirements placed on the communication system are stringent.

- Physiological and other data is communicated real-time between non-deterministic endpoints. One data source is always replicated to many subscribers which changes dynamically.
- Communication between a producer and the multitude of subscribers **cannot** be considered inherently reliable. Subscribers may be affected as a whole, or individually, and each subscriber should be "caught-up" with any data that was lost due to a communication failure (up to hours of lost communications.)
- The clinical monitoring configuration of a patient… devices associated, demographic and physiological attributes, and algorithm operational parameters… can be modified anywhere in the ecosystem, and a core critical set of the patient state should be allowed to be modified **even during communication system failures**. When communication is restored, the patient state is safely reconciled between all participants in the ecosystem.
- The communication payload is dynamic, as physiological parameters are added and removed from the patient, or new algorithms are activated.
- The communication messages must be encrypted and authenticated. Devices must authenticate with each other. In addition, an authorization scheme must be present to allow users and devices to subscribe and publish changes to the system state.
- The communication payload must be efficient since many communication links are wireless and the devices are battery operated.
- The communication system must scale to 100,000s of communicating endpoints.
- As the software is upgraded with enhancements, communication must continue to operate as the system is in a state of transition… it will forever be in a state of transition.
- The communication system should have no single point of failure. Device/service level failure protection is not enough; geographically failure must also be considered. The loss of a datacenter due to a catastrophic event (earthquake, explosion, etc.) is possible, and likely a situation where a surge of additional patient monitoring is needed … Patient monitoring must continue.
- The system should integrate onto a converged customer IT infrastructure, supporting modern IT technologies and architectures.

IV. Why DDS

The Data Distribution Service standard was specifically designed for real-time, mission-critical applications to manage data-centric states across decentralized systems, in a scalable and secure manner.

DDS is not "cloud centric". Like many message broker technologies (i.e. Kafka), there is no centralized cluster of services to receive and forward all messages. From a patient monitoring perspective, this is essential, so that devices can communicate directly with each other and not be dependent on reaching a centralized service. Considering the potentially large footprint of a single patient monitoring system (over 1M sq. ft.) it is incredibly important that we engineer the *geographical databus*, leveraging DDS routers, taking into account failure modes and application needs.

With direct, device-to-device communication, information is exchanged at “wire-speed” with Quality of Service control… prioritizing physiological critical events over historical patient data, as an example. DDS is designed to work in a dynamic, chaotic environment, with publishers and subscribers continually participating and leaving. As uncomfortable as this sounds, because of the dynamic nature of a very large patient monitoring system, often data states are “eventually consistent”.

Eventual consistency is a paradigm we must embrace, requiring a technology that continually, and correctly reconciles state in a predictive and deterministic fashion, remaining clinically safe and effective. As an example, if a hospital care unit is “cut-off”, due to a variety of communication infrastructure failures, from a centralized set of service (i.e. a data center), the patient acquisition devices must continue to communicate with the unit’s central station. For the system to continue to be effective, clinicians must be able to modify the state of the isolated unit; continuing to admit, move and discharge patients, and adjust alarm limits locally during the infrastructure outage. In this “split brain” scenario, the isolated unit’s state has changed independently, and when the connectivity is restored, the entire system reconciles and becomes consistent… eventual consistency. In this example we leverage the distributed state of DDS to our advantage. There are no data-mastership concerns, there is just distributed state reconciliation.

Also, to reduce latency of communication, we needed a solution that *pushes* data state changes across the ecosystem. Many common message brokers require the client to poll the server periodically for new messages (i.e. Kafka); this element is a fundamental differentiator between true real-time systems that exchange the current state and a typical message broker. Again, DDS is fundamentally designed with this type of performance in mind. In patient monitoring, physiological waveforms (i.e. electrocardiograms, plethysmographs, arterial blood pressures, electroencephalograms) are treated as a rapidly changing data states of signal acquisition devices.

In addition, with the dynamic nature of our payload, and the requirement of efficient communication, DDS has a built-in “type system” (DDS-XTYPES) that provides the language for “type definitions” managing compatibility and discovery. This allows our data model to change dynamically and evolve over time while maintaining backward compatibility. This tool to manage data-model compatibility is a significant element to solve the problem of a “system in constant transition”. Since DDS understands the data types, it allows for efficient data distribution through content aware filtering (even at the source), queries on content, and context-aware forwarding of data through the DDS router. These are extremely powerful capabilities to make the system efficient. Again, most messaging systems only understand Key/Value tuples, which puts the burden on the producers and consumers to serialize and deserialize into a data model, as well as the burden of managing data model changes. Key/Value tuples become significantly less efficient, with very large communication payloads, as the data model grows. With very complex data models derived from human physiology, it is absolutely necessary that the data distribution system has strongly-typed data-model adherence across the system and is efficient as the data model expands.

And finally, some DDS vendors provide a simple add-on solution to not only compress the content of the message, driving further system efficiency, but also with DDS built-in functionality fully secures the messages. DDS domains enable logical segmentation of different application communication patterns, allowing for the management of information flows between endpoints to further drive efficiency and scalability. Since DDS is transport independent, the best transport is used for the communication link… let it be TCP, UDP, multicast or even a custom wireless serial link (which are used in wireless patient monitors). The transport choice does **not** need to be homogenous throughout the system.

V. Challenges

DDS is extremely powerful, which makes it a complex technology to use correctly. It requires expert tailoring to fit specific applications. It would be naive to believe that complex problems, such as our patient monitoring ecosystem, can be solved with simpleton solutions. Many messaging technologies target simple needs, and ultimately, if chosen for a complex problem, push the complexity to a different part of the system design. The DDS challenge comes in the form of building internal expertise, patience, applying methodical system design and testing discipline, and more and more testing.

The model we employed at GE Healthcare Patient Monitoring consists of a core of experts who develop a DDS endpoint that is provided to engineers who design the functionality of the patient monitoring system. This provides some abstraction from the design of the physiological acquisition devices, clinical viewers, clinical algorithms, and external system interfaces. All the teams have basic DDS knowledge and awareness but are relatively abstracted from DDS complexities. The teams understand the API and service level provided, and use the resulting in-memory data model, event hooks and methods to handle state changes and pushing updated data states.

Illustration of caregivers in a clinical unit with one clinician at the central station collecting clinical information on one of the patients monitored by the system. Includes real-time review of multiple patients on the central station, a wall-mounted display and a hand-held device.

VI. Conclusion

At GE Healthcare Patient Monitoring, we are extremely proud of the patient monitoring and therapy system foundation we've built and are continuing to expand upon. We know that one day our friends, family and even ourselves will be a patient on the system. We carry that responsibility with us each day. We know the high stakes of failure. We utilize the correct and best technology to solve this extremely complex and important challenge to help ensure that patients are safely monitored.

VII. About the Author

Matthew Grubis is a Chief Engineer at GE Healthcare's Patient Monitoring business. Matt's extensive experience in patient monitoring spans 22 years and he has worked with 100's of healthcare institutions from around the world. Matt has his Bachelor of Science in Electrical/Computer Engineering and Computer Science from the University of Wisconsin – Madison, as well as a Master of Science degree in Electrical Engineering with an emphasis on communication systems, signal processing and information theory from Marquette University in Milwaukee, WI USA. Matt also serves on the board of the DDS Foundation as the Vice President of Applications.

GE Healthcare Approved Document JB76285XX(1)