Design Pattern: Many to One

TYPICAL APPLICATIONS

Distributed Sensor Networks

Industrial Automation

Mobile Asset Tracking

Overview

There is a common use case in distributed applications where many devices or sensors can be fielded over a wide area all producing the same type of data. A typical example of this data is temperature data or alarm sensor data or even a generic I/O data point. In distributed publish/subscribe architectures, this is known as a Many to One distribution pattern.

Design Considerations

There are many aspects to consider when architecting a system that includes this design pattern. Depending on the scale of the system, the amount of Data Distribution System (DDS) discovery can be limited by configuring the sensors to only discover the appropriate subscriber. Also, if the load on the subscriber is too high, then the number of subscribers can be increased to accommodate the appropriate amount of publishers.

Figure 1 shows a system where there are many sensors that are all publishing data of the same type. The SensorType listed here is just an example datatype that can easily be adjusted to accommodate many different kinds of data. The important characteristic to note here is that one of the fields, sensorID, is listed as a "Key" field. By setting up your datatype with a "Key", any application that subscribes to the Sensor Topic can now easily distinguish between which sensors are sending the data.

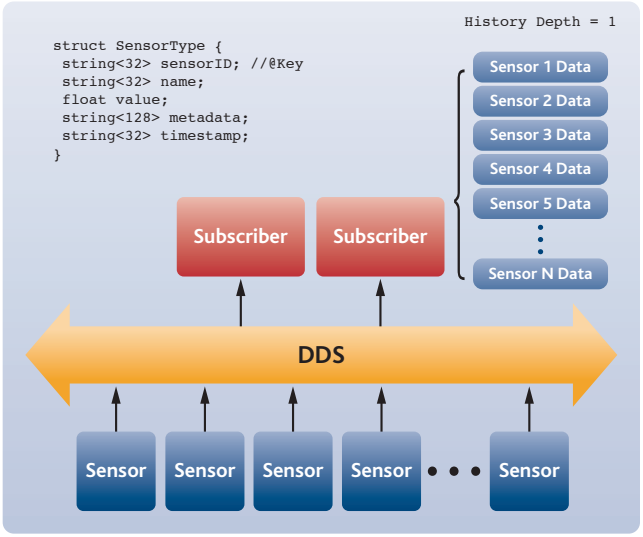


Figure 1.

This coordination of separating incoming data from many sources is provided directly within the DDS infrastructure so that your applications can easily coordinate what is happening with each of your sensors or publishers of data. Also, by setting up a subscription to a topic with a key defined, enables your application to access their data locally without the need to request each sensor to send its data. If a sensor becomes inactive, the knowledge of this event is now available to the subscribing application on a per instance basis. This configuration now provides a completely dynamic architecture where sensors can come online at anytime, their operational status can be determined easily by the subscribing applications and the data of many sources of information can be easily read and distilled in a single location.

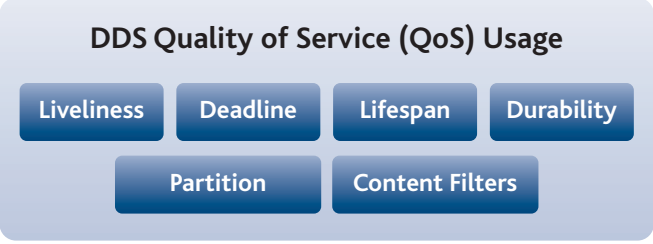
Applicable DDS Quality of Service Parameters

Liveliness (DataWriter)

By setting up liveliness on the DataWriter of each sensor with a particular time period, each subscribing application can now be notified at the end of this time period that a particular sensor is no longer alive.

Deadline (DataReader, DataWriter)

If your sensors are periodic sensors that provide new data at a given rate, the deadline time period QoS enables a contract with subscribers to be setup that will declare your minimum publication rate. If this rate is not met, DDS will notify your subscribing applications that its deadline has been missed. Also, if there are subscribing applications that require data faster than what your defined deadline rate is, then DDS will guarantee that these applications do not communicate because the subscriber is asking for data faster than the publisher has declared it can deliver.



Lifespan (DataWriter)

Each sensor can also specify how long its new data is to be considered valid data. This time period is then utilized by the infrastructure to insure no data is accessed if its Lifespan time has expired. DDS accomplishes this task by removing all data from DataReader and DataWriter history queues at the expiration time. Using this QoS the subscriber application can be ensured that when it reads data from its local DataReader history, its data is not stale.

History (DataReader, DataWriter)

Setting history to a given size on the DataReader ensures that each instance of the Sensor-Topic will have that amount of local historical data. Each subscribing application can setup its own size of history. Therefore one subscribing application can setup a history depth of 1 meaning it only has the latest value of each sensor, while another application can setup a history depth of N where it can then enable easy trending analysis to occur.

Durability (DataReader, DataWriter)

By using a Durability setting of Transient Local, each Sensor can be setup to hold its latest N data samples such that new subscribing applications can automatically be sent historical sensor data without having to perform any request. DDS takes care of delivering durable historical data automatically to all new DataReaders of that topic.

Partition (Subscriber, Publisher)

If there are a large number of sensors or publishing entities, then applications may want to segment groups of sensors for coordination. For example, each section in an industrial automation plant might setup its own partition group. When doing so, DDS makes it easy for subscribing applications to designate which group, set of groups or all groups of sensors it would like to receive data from. And these group designations can be reconfigured dynamically to enable applications to quickly change their particular region of interest.

Content Filtering (DataReader)

Each subscribing application can further filter which data it wants to receive by specifying an SQL-like filter clause. This filter enables an application to only receive data from sensors when its value is above a particular threshold or maybe in a specified range of values. Filters could also be setup to only look for error conditions or alarm conditions. Content filters can be applied any of the fields in the topic and they are dynamically changeable so that applications can easily change their region of interest. Also note that Content Filters can be executed at the actual sensor so that bandwidth on the network is being optimized.

U.S. HEADQUARTERS
Real-Time Innovations, Inc.
385 Moffett Park Drive
Sunnyvale, CA 94089
Tel: (408) 990-7400
Fax: (408) 990-7402
info@rti.com