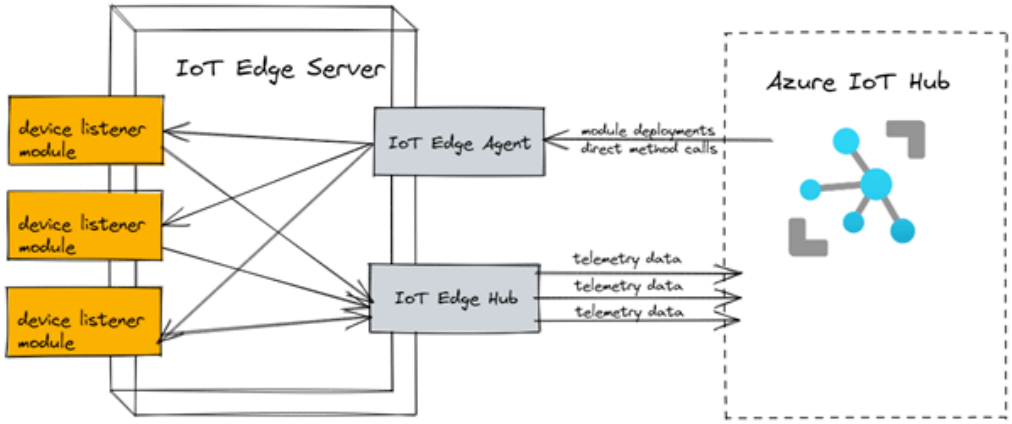
FUJITSU

Improving Azure IoT Edge application robustness

IoT solutions differ from other software and data platform solutions in that they need to be able to quickly recover from failures, reduce the amount of network traffic and be able to deal with being offline for a period. Adding an Azure IoT Edge server into the IoT solution can help implement some of these design features. But it needs to be configured properly for offline support and can be further enhanced by implementing a circuit breaker design pattern.

What is the Azure IoT Edge Server?

The Azure IoT Edge server is a feature that allows telemetry devices that do not have native internet support, to connect to the Azure IoT Hub via a server device. The IoT Edge server also allows for analytics to occur close to the source. For example, a machine learning model that alerts an operator of a predicted event. Finally, the IoT Edge server can also be used to reduce the amount of bandwidth costs of data movement, by being able to filter and aggregate near source prior to message creation. The Azure IoT Edge server exists as a runtime with some software modules that are deployed in a containerized format along with custom built modules.



Included in the IoT Edge runtime are the IoT Edge Agent which manages the module deployment and monitoring, as well as the IoT Edge Hub which manages communications with devices and the IoT Hub.

Custom built modules can be developed to interface with the non-internet connected sensors to listen for incoming telemetry data. These modules can combine telemetry data from several sensors, aggregate it, compress, and transform it into a protocol compliant format to ease transmission.

Why does the IoT Edge Server require robustness?

Being a container format, the IoT Edge server must share its finite resources across the various container modules. Each container module acts like its own virtual machine with just enough bare bone resources to keep it running. This competition for resources as well as other factors, can result in modules becoming non-responsive from time to time.

Another factor to take into consideration is that the IoT Edge server may not be constantly connected to the internet. For example, an IoT Edge server could be located within a black box on a moving vehicle such as a locomotive that travels across deserts, through mountain tunnels and other remote locations.

How can an IoT Edge Solution be designed for robustness?

Configuring the IoT Edge Agent

To improve robustness, the IoT Edge Agent is configured to monitor the listener modules continuously. The agent can be configured to respond to particular status messages as well as time outs. Typical responses to failed statuses or timeouts is to restart the module.

Design the Edge Modules for High Availability

To facilitate High Availability, modules should be designed to reboot as fast as possible. They should not rely on connectivity, particular resources becoming available or remote libraries. They should be designed to fail gracefully. A failed module run should not impede successive runs.

All exceptions should be handled, and any shared artifacts must be reset upon exception capture. Restarts should not be attempted inside the module but implemented in line with the IoT Edge Agent configuration.

Design the Edge Modules for Offline support

The Azure IoT Edge can be configured for Offline support. In fact, after the initial handshake with the IoT Hub, where device twins are shared (including security identities), the IoT Edge could run in offline mode indefinitely. But this is not a feasible design for an IoT Edge Server.

There are several offline support features that can be configured to work with the IoT Edge modules:

- When offline, the IoT edge server can be configured to save off any unsent messages into local storage
- Can act like a de-facto IoT Hub to authenticate child devices (when a device hierarchy has been implemented
- Can facilitate communication between various modules

Things to take into consideration for offline mode include the total amount of storage capacity available to the server, and the time to live (TTL) setting for the telemetry messages. In offline mode, messages will only be stored until their create timestamp passes the TTL threshold. Increasing the IoT server device's storage means that more messages can be stored.

In order to optimise the IoT Edge during offline mode, first considerations should be made into how much physical storage can be added the server boxes. Also, an analysis of the size requirements for the telemetry messages should also be taken into consideration. Finally, an analysis into the number of containerised modules and their resource requirements also need to be undertaken.

Once all these things have been taken into consideration, the TTL and the amount of storage allocated to the IoT Edge module should be configured for optimal robustness.

Further to these configurations, there is another feature that is available to IoT Hub that can also be conditionally included in the module functionality. IoT hubs have the facility to accept file uploads from the IoT Edge. This opens the possibility for archived messages to be pushed directly into cold storage if they have surpassed a particular threshold.

A circuit breaker design can be implemented to improve robustness utilising these offline capabilities.

Implement a Circuit Breaker Design pattern

The circuit breaker is a design pattern, popularised by Michael Nygard in 2018, that can prevent an application from repeatedly calling an operation that it likely to fail. In the scenario mentioned previously, where the IoT Edge server temporarily loses connectivity to the internet, the circuit breaker pattern can be implemented to help optimise the offline support options.

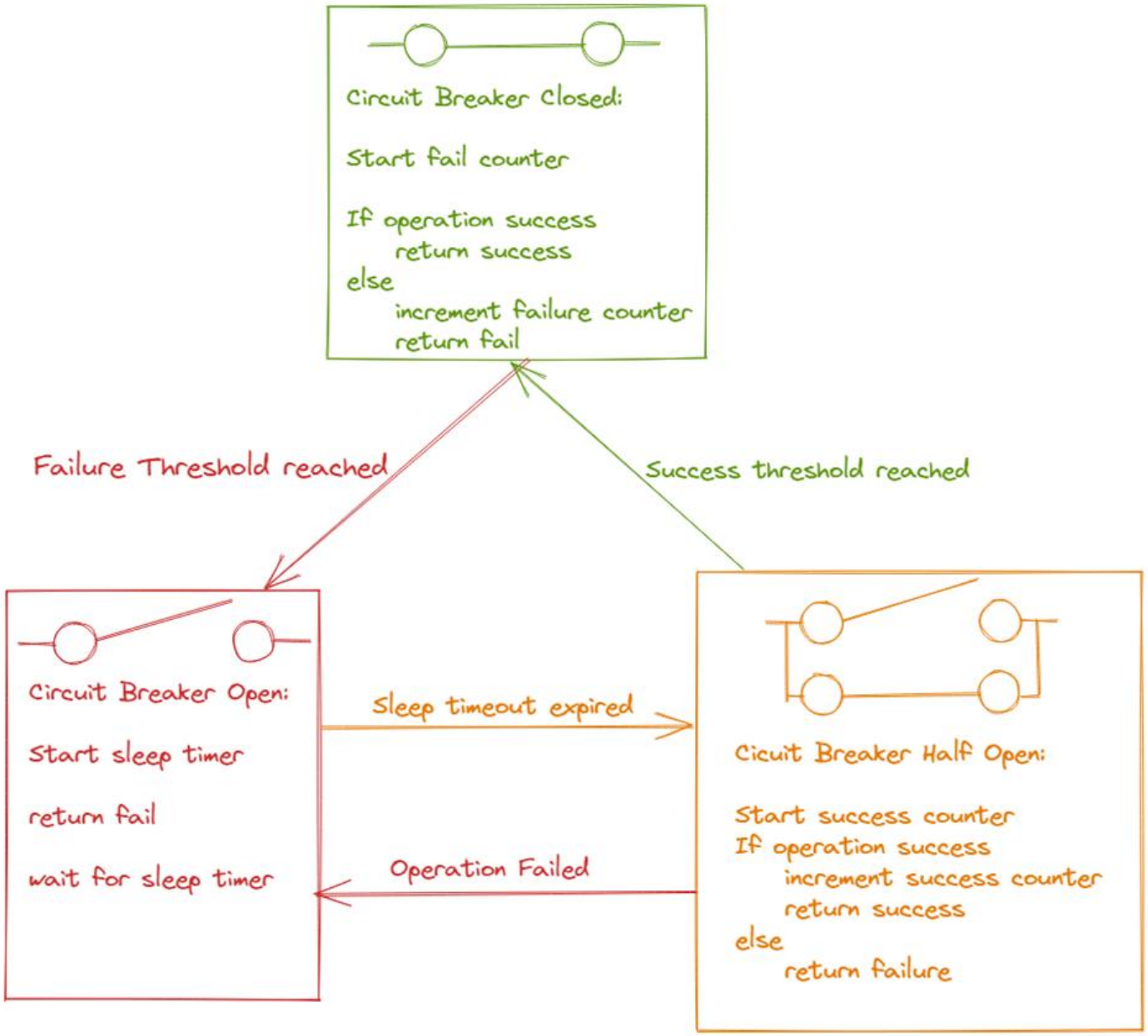
The physical boxes that host the IoT Edge runtime have finite resources, including storage capacity that is shared amongst all the containerized edge modules.

At some point in time in offline mode, with telemetry data continuously flowing in, the storage capacity will become depleted. A trade off needs to be made.

Does the module simply purge the older messages?

Or is there an alternative?

In the scenario where an IoT Edge server becomes offline, a circuit breaker pattern can be implemented, to reduce the number of concurrent calls to the IoT Hub, and prepare the module for an alternate communication approach.



After the failure threshold has been breached, the circuit breaker pattern is in a closed state. A sleep timer is set where no calls are made to the IoT Hub. In this state, the older messages are converted into compressed files that reduce storage footprint needed to store the individual JSON messages. The older messages are then purged.

After the sleep timer has expired, the circuit breaker goes into a half open state. Here the operation is re-tried with a much lower traffic volume to normal. The initial calls to the IoT hub can be the file upload of the compressed files. If these succeed, then the success counter is incremented, and more compressed files can be uploaded. After all the compressed files have been uploaded, then the module can switch back to sending the individual messages in FIFO order.

As soon as the success counter has exceeded the defined threshold, the circuit status returns to closed and messages continue to be sent to the IoT hubs as individual packets.

Conclusion

The Azure IoT Edge offers a good solution for connecting non-http enabled sensors to an Azure data platform. IoT Edge modules need to be designed with robustness and offline support in mind. With finite resources available on every physical box that runs the IoT Edge runtime, Edge modules need to be carefully designed to be as resource efficient as possible.

A circuit breaker design pattern can be incorporated during offline mode to switch the processing focus to resource preservation. Local storage can be optimised and the risk of data loss can be reduced.

If you need help implementing robust IoT edge solutions at your organization, please contact our Fujitsu Data & AI specialist now.

Contact

Fujitsu Data & AI

+61 3 9924 3000