Technical Disclosure Commons

Defensive Publications Series

July 2020

IOT OAM CONNECTIVITY MODEL - END-TO-END APPLICATION LAYER RELAY PING FOR CONNECTIVITY CHECK

Nagendra Kumar Nainar

Carlos M. Pignataro

Akram Sheriff

IOT OAM CONNECTIVITY MODEL - END-TO-END APPLICATION LAYER RELAY PING FOR CONNECTIVITY CHECK

AUTHORS:

Nagendra Kumar Nainar
Carlos M. Pignataro
Akram Sheriff

ABSTRACT

Methods are provided that allow for detecting the connectivity between an Internet-of-Things application (IoT-App) and a device at the application layer. In one method, an IoT gateway (IoT-GW) acts as a proxy that triggers a Request message to the device upon receiving a proxy request from the IoT-App with relevant details such as correlation identifier (ID) that can be used to correlate the response to the request. In another method, local statistics cached by the IoT-GW are leveraged for each device.

DETAILED DESCRIPTION

In an Internet-of-Things (IoT) environment, a MQTT local broker (bridge mode) runs on IoT gateway (IoT-GW) that receives data from multiple devices/sensors using MQTT or similar protocols. (MQTT is a lightweight messaging protocol for small sensors and mobile devices, optimized for high-latency or unreliable networks.) An IoT application (App) running in the cloud will consolidate the telemetry data from different such IoT GWs. The MQTT Pub/Sub topic based messages are used for the data communication between IoT-App and IoT-GW, as shown in Figure 1 below.

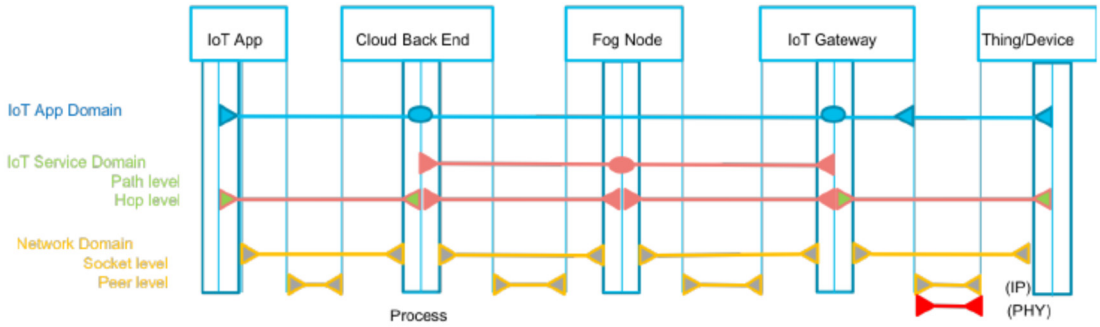

Figure 1

While existing Operations Administration and Maintenance (OAM) for connectivity checks functions at Layer 3, it is not sufficient to validate the connectivity check at Layer 7 between the IoT-App and the device/thing. In other words, Layer 3 connectivity between a Client/App and broker does not guarantee that the application layer messages are successfully exchanged. It is therefore not possible to ensure that the data sent by the device/thing is received and forwarded to IoT-App in the cloud.

Presented herein is a mechanism for an application layer proxy ping that can help to detect the connectivity at the application layer between the device and the App.

A new proxy ping message is introduced as part of IoT transport protocols such as MQTT. While these concepts are described with respect to the MQTT v5.0 spec, the same can be extended for other protocols as well.

Embodiment 1:

The IoT-App from the cloud sends a proxy ping to the IoT-GW that is handling multiple devices to which the connectivity check is performed. The proxy ping carries the following details:

Device-ID --> To which the connectivity is performed.

Correlation ID --> Used to map the response received to the request sent from cloud IoT-App.

Reply-To Topic --> Topic to which the response is to be published.

The IoT-GW, upon receiving the same, will generate a Request message with the above details and forwarded it to the device. The device will publish the data to the topic using the received correlation-ID. This will be forwarded towards the IoT-App. The IoT-App receiving the response will use the correlation ID to match to the request sent.

Embodiment 2:

The IoT-App sends a proxy ping to the IoT-GW that is connected to the device to which the connectivity check is performed. The proxy ping carries the following details:

Device-ID --> To which the connectivity is performed.

Correlation ID --> Used to map the response received to the request sent.

The IoT-GW will check the local cache for the last timestamp that a message is received from the device. The IoT-GW will reply back with the last received timestamp in the response to the cloud based IoT-App (or the requestor). The IoT-App receiving the response will use the correlation ID to match to the request sent.

The IoT-GW may not natively support MQTT or an OAM agent. In such a case, the OAM agent may run as a virtual device on the IoT-GW (as a container for example).

Container software may probe IoT-GWs on application platforms that would directly communicate with a network controller entity using gRPC (Remote Procedure Call) protocol. Such a virtual probe based OAM agent may run on the IoT-GW software in different hardware platforms.

Embodiment 3:

In cloud-based IoT-Apps, there is a concept of "AWS Device shadow" of storing the shadow information of all devices in a Javascript Object Notation (JSON) file in the cloud (IoT-App) that maintains a DeviceID : datastore mapping about the persistent state of the different IoT Sensor/devices connected to the IoT-GW. If there is a network connectivity issue between the IoT-GW and IoT-App, then by using the "update/delta MQTT topic", the IoT-App can inform about the desired state of IoT device via the IoT-GW when the network connectivity resumes. However this approach does not consider the Layer 7 MQTT App ping failures when the IoT devices connected to the IoT gateway over low power wireless links like LoRaWAN or 6LoWPAN would experience in a real IoT network.

This solution is based on the MQTT v5 spec defined in https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html#_Toc3901252, and specifically, section 4.10.

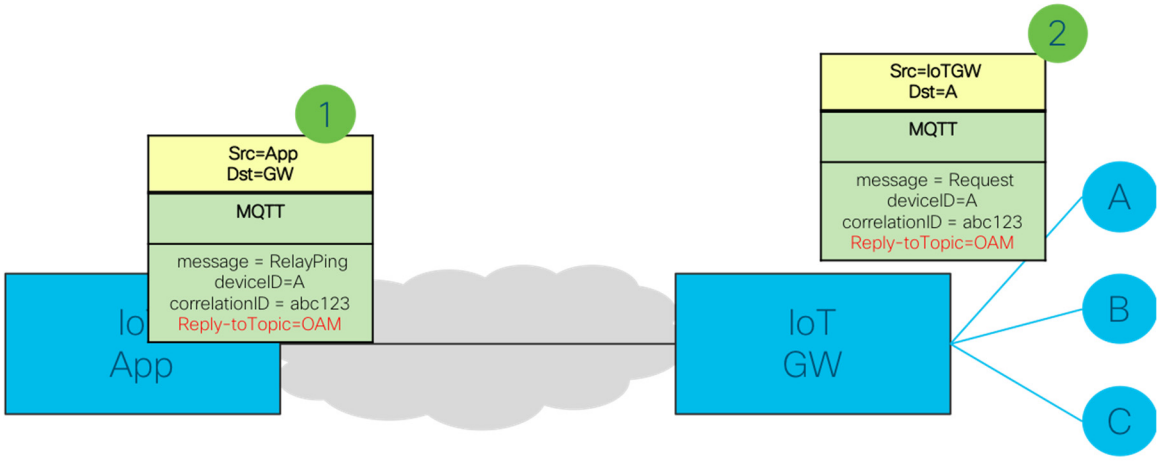

Figure 2

As shown in Figure 2, the IoT-App will send a new message type as "RelayPing" to the IoT-GW. The semantic associated with this message is to trigger a Request message to the DeviceID (that is received in the RelayPing). Accordingly, the IoT-GW upon triggering the Request message, will generate a Request message and copy the details such as DeviceID, CorrelationID and Reply-to-Topic to the Request message. This message will be sent to the Device. The message is sent to Device A.

The device will generate the relevant data and publish the same to the Reply-to-Topic along with the CorrelationID as defined in the received Request message. This leverages the existing mechanism defined in MQTT v5 and so it does not need any new message or enhancement in the device.

The IoT-App will use the CorrelationID from the received data (published by the device to the topic requested by the App) to identify if the response is received. This helps to detect the connectivity at the upper/application layer.

In another variation, the IoT-GW leverages the local cache that maintains the flow statistics from the devices connected.

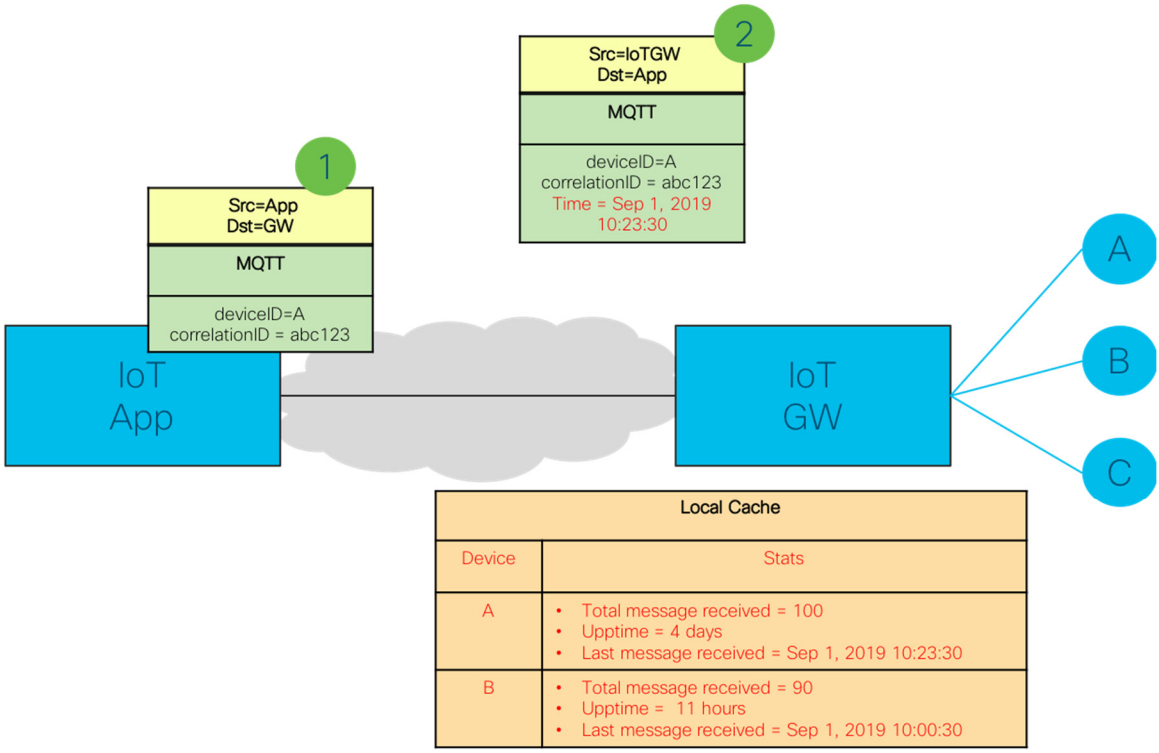

Figure 3

In the Figure 3, the IoT-GW will maintain the last received timestamp for each device. The IoT-App will send a new message type to IoT-GW carrying the following details:

Device-ID (or list of Device IDs)

Correlation ID

The IoT-GW will use the local cache to fetch the relevant data, such as the timestamp of the last received message from the device. The response will carry the details to the IoT-App.

The IoT-App uses the local data (last received message from DeviceID A) and the timestamp from the IoT-GW. There may be some difference in the timestamp due to the time synchronization, delay in parsing the message/data from device, etc. The IoT-App considers the above delay (like a threshold) to ensure that the timestamp is within a certain threshold to consider it as a success.

The MQTT Layer 7 application binding with the network connectivity state of an IoT Client/sensor is important from the perspective of the MQTT broker running on IoT-GW. This is because IoT clients connect to the MQTT broker running on the IoT-GW and subscribe to or publish data on specific topics. The broker is in charge of forwarding the data published to the clients interested in it, thus decoupling the process of data generation and consumption both in space and in time. Standard MQTT specifies a Keep Alive parameter that defines the maximum time interval permitted to elapse after the last client transmission. In case the timer expires, the broker closes the connection with the client. Therefore, to maintain the connection alive, a client periodically transmits PINGREQ messages. If for some reason the 6LLN or other LPWAN IoT clients lose the network connectivity with the IoT-GW where the 'MQTT Broker' is running, then currently, per the MQTT 5.0 spec, there is no mechanism for the cloud based IoT-App (management application) to reach the IoT client/endpoint in a reliable and timely manner. As a consequence the cloud IoT-App may even store stale data about an IoT client/sensor. To avoid this specific problem at the cloud application level, a change is made to the state/entry of the IoT client in the MQTT broker running on the IoT-GW. Since the IoT-GW maintains the L2/L3 table entry of an IoT client as per different wireless technology (BLE, 6LLN, CG-Mesh, Wi-Fi, LoRaWAN, etc.) used, the MQTT broker running on the IoT-GW at application layer/Layer 7 has to auto-trigger this state change at Layer 2/Layer 3 automatically provided the MQTT broker does not receive the keepalive messages from the IoT client. This results in a new auto-trigger functionality from the Layer 7 to Layer 2/Layer 3 binding with an MQTT application layer based Relay-ping message.

There are advantages to this proposal. First, if for some reason an IoT client/sensor associates to another adjacent/neighbor IoT-GW where the MQTT broker is running, the former/old MQTT broker's Layer 7/Application layer relay ping provides a counter-based mechanism if MQTT-based traffic for that specific client is forwarded in both ways (IoT Client--->IoT-GW--->Cloud IoT-App) and (Cloud IoT-APP--->IoT-GW--->IoT Client). If NOT, then a trigger is made for: a) Layer 7 relay ping based state change in the old IoT-GW where the MQTT broker is running; b) Trigger the Layer 2 relay ping among other neighboring IoT-GWs where the MQTT broker is running; and c) identify to which

neighbor IoT-GW to relay the MQTT response coming from the cloud IoT-App to the IoT Client.

Second, this proposals involves triggers the replication of traffic from the old IoT Broker to the new IoT Broker where the IoT Client has moved to associated state. This provides an MQTT application layer---> Layer 2/Layer 3 based state replication among MQTT brokers running on different IoT-GWs as well as robustness against failures in network connectivity. This is very useful in roaming scenarios to have MQTT- Layer 7 application relay PING Req/Response based auto-trigger for state change and then relaying the traffic to the appropriate IoT-GW to which the IoT client has roamed, such as in transportation, high speed Metro, and fleet management IoT use cases.

As another variation, in a distributed MQTT broker stack scenario with MQTT brokers running in different IoT-GWs that are clustered/grouped, this method has the value to optimally relay the MQTT application layer PING Req/Response only to that specific MQTT broker that has the state of the IoT client associated in real-time and which can publish an message to an MQTT Topic.

In summary, two methods are provided that allow for detecting the connectivity between an IoT-App and device at the application layer. In one method, the IoT-GW acts as a proxy that triggers a Request message to the device upon receiving a proxy request from the IoT-App with relevant details such as CorrelationID that can be used to correlate the response to the request. In another method, the local statistics cached by the IoT-GW are leveraged for each device.