Notes on MQTT-SN v.2.0 specification WD 17

Miroslav Prýmek
miroslav.prymek@energomonitor.cz

May 2022

Abstract

When implementing Bisquitt, the transparent MQTT-SN gateway, we discovered several weaknesses of the MQTT-SN v1.2 specification and faced some uncertainties about the right specification implementation. Some of the problems were addressed in the v2.0 draft, some still persist.

This document describes supposed bugs, uncertainties, and our suggestions for specification improvement. I am not a member of the MQTT-SN subcommittee and don't have access to the internal discussion, hence some of the comments can be wrong just because I don't have the right context or I just didn't read the specification well. Thank you for your tolerance.

1 Bugs

1.1 "Aliases exhausted" reason code missing

The mumber of topic aliases is limited to 65534. If a client/gateway attempts to register more than 65534 topics, a gateway/client should probably return `REGACK` with a specific return code because the reason codes specified in [2.4.3 Reason Code] are inappropriate or have different meanings in this situation.

1.2 `WILL*UPD` is hard to implement right in a transparent gateway

MQTT v.5.0 doesn't have any equivalent of `WILLMSGUPD` and `WILLTOPICUPD` messages. Will topic and message are part of the `CONNECT` message. Hence, if the gateway should update the will topic or message, it must close the connection and open another one.

According to my understanding, the transparent gateway should open the connection to the MQTT broker only whenever it receives an MQTT-SN `CONNECT` message and keep it open until MQTT-SN connection is closed. Closing the connection because of a will topic/message update would IMHO decrease the robustness of the gateway: if it crashes between connection closing and reopening, the client is lost but *the will message is not sent*.

1.3 Wildcards in a short topic

> *Note however that it does not make sense to do wildcarding in subscriptions to short topic names, because it is not possible to define a meaningful*

name hierarchy with only two characters.

[4.10 Pre-defined topic alias' and short topic names]

Actually, there can be `/+` and `/#`. It is a bad practice to start a topic with a slash but some people do it and the MQTT 5.0 spec does not forbid it:

Topic level separators can appear anywhere in a Topic Filter or Topic Name.

[4.7.1.1 Topic level separator]

MQTT-SN spec should probably explicitly forbid or allow the usage of wildcards in short topics.

1.4 UTF-8 strings length field

Each of these strings is prefixed with a Two Byte Integer length field that gives the number of bytes in a UTF-8 encoded string itself [...]

[1.7.4 UTF-8 Encoded String]

I think this is a bit misleading because the definition looks like *all* strings should be prefixed by their length. But most of the string fields are not because their length is implicitly given by the packet length (e.g. `CONNECT`, `WILLTOPIC`, `REGISTER`).

1.5 Confusing `AUTH` example

In 4.19 Authentication, the non-normative example:

1.5.1) the `CONNECT` packet should not contain the *Authentication Data* field

1.5.2) "rc=0x01" is wrong (value not defined in Table 23)

1.5.3) the last row: "Gateway to Gateway" → "Gateway to Client"

Also, it would be nice if the example described a continued authentication:

Client to Gateway: CONNECT Authentication Flag=1
Client to Gateway: AUTH rc=0x0 Authentication Method="PLAIN" Authentication Data=client-first-data
Gateway to Client: AUTH rc=0x18
Client to Gateway: AUTH rc=0x18 Authentication Data=client-second-data
Gateway to Client: CONNACK rc=0

1.6 Predefined aliases known beforehand by both parties or not?

If the client initiates a `REGISTER` against a topic which is known by the gateway to have a predefined topic alias associated with it, the gateway will specify its topic alias type to be predefined and set the topic alias value to match this in the `REGACK`. The client can then choose to update its registry of predefined topic alias' if it so wishes.

[4.8 Topic Name and Topic Filter Registration Procedure]

This means that the predefined alias can be known by the gateway and not by the client. I'm not sure that this is a good concept. I always considered predefined aliases to be a *read-only* map of aliases known by both parties before a connection is established.

If the client *must* expect that for any `REGISTER` it can be given back a predefined alias, then the predefined aliases map must be read-write as well as the "normal" aliases map, but still these two must be separate. This can be inconvenient for a constrained client because it would have to allocate two big enough read-write maps or use one map but tag every alias as predefined or normal.

This, from my point of view, makes the whole concept of predefined aliases almost useless (because they can't be hardwired and/or read-only).

Anyway, the above-cited paragraph conflicts with the definition

> *A predefined alias is known statically by* both *the gateway and the client outside the scope of a gateway session. No negotiation is required since* both *entities have knowledge of the topic alias mapping.*
>
> [2.5 Topic Alias Types]

1.7 Sleep is hard to implement in a transparent gateway

> *During the AWAKE state, for each packet the gateway sends to the client, the application messages' quality of service shall be honoured, and a full packet interaction shall take place including all normative phases of acknowledgement, including any associated retransmission logic.*
> *The transfer of packets to the client is closed by the server/gateway by means of a `PINGRESP` packet.*
>
> [4.18 Support of sleeping clients]

Because the gateway "closes" the *Awake* state with a `PINGRESP` packet, it must introspect, understand and track all the flows taking place in *Awake*. It can send `PINGRESP` only after all of them are finished. This is a little bit hard to implement especially for a transparent gateway, which can react to some of the packets strictly reactively (without tracking the flows, leaving it to the MQTT broker).

It would be more convenient if the *Awake* included only sending of packets buffered on the gateway. The gateway would just send the whole packet buffer out and then close the session with `PINGRESP`. If the client wanted to react somehow, it could do it in another "window" or switch to the *Active* state.

I know this would conflict with the "just one flow of a kind" rule (3.3) and these two must be considered together.

1.8 Harmful frequent aliases re-registration for sleeping clients

> *Topic Alias mappings exist only while a client is active and last for the entire duration of the active state. Therefore, the gateway must re-register*

any topic alias's during the AWAKE state, which will last until the last PINGRESP is issued.

[4.18 Support of sleeping clients]

I consider this a very unfortunate decision. It's vital for low-power devices to keep the awake period as short as possible. For clients which sleep most of the time and wake up periodically just to send/receive messages (IoT sensors, etc.), the alias registration should really be preserved between *Awake* periods. If the client had to re-register aliases with every wake-up, it could send the whole topic name within the `PUBLISH` message like in MQTT — i.e. in this scenario, the very reason for existence of MQTT-SN is contradicted, I'm afraid...

This effect is even worse with the "at most one outstanding `REGISTER`" rule (see 3.3) because if the client must wait for the reply before sending another `REGISTER`, the number of aliases registrations are effectively multiplied by the network roundtrip time (at best) or even $T_{retry} * (N_{retry} - 1)$ which would be devastating for low-power devices like battery-operated sensors, etc.

Also, in Figure 5, the `REGISTER` and `REGACK` packets are not drawn.

1.9 Other

1.9.1) *A topic alias is a 16 bit integer* [1.3 Terminology] → 16 bit *unsigned* integer (?)

1.9.2) an *UTF-8 String Pair* is defined in 1.7.6, but not used anywhere

1.9.3) a `DISCONNECT` *Duration* field is mentioned multiple times, but it was renamed to *Session Expiry Interval*

1.9.4) in 3.1.24.4 Reason String, the string encoding should be defined (UTF-8?)

1.9.5) in 3.1.29 Forwarder Encapsulation and 4.1.2 Aggregating Gateway, the "WSN" abbreviation is used and not defined anywhere

1.9.6) Table 33 seems a little bit misleading to me: *Topic Filter* is an alternative to *Topic Alias* here but the horizontal axis describes bits of a byte (at the first sight it looks like the *Topic Filter* should occupy bits 3..0). Also, it looks like the length of the *Topic Filter* should be 2 bytes. The description says correctly it's "6..N" but it is not clear enough visually IMHO...

1.9.7) Table 38: "publish to gateway", "received a `PUBLISH` packet" → subscribe to topic, `SUBSCRIBE` packet

1.9.8) Table 38: title "`SUBACK` Reason Code" → `UNSUBACK` Reason Code

1.9.9) *the mapping table between registered topic ids and topic names* is *implemented per client* [4.9 Topic Name and Topic Filter Mapping and Aliasing] → *MUST* be implemented (?)

1.9.10) the removed *CleanSession* flag is referenced multiple times

1.9.11) *Like MQTT, sending the* `DISCONNECT` *packet does not affect existing subscriptions and Will data if the CleanSession flag is set.* [4.16 Client's Disconnect Procedure] — this is unclear/misleading

1.9.12) *see Section 4.17 for further details* [3.1.22.3 Client Identifier (optional)] → 4.18 (twice in this section)

2 Underspecified

This section describes situations that can happen in practice and I can't tell from the specification what the right behavior should be.

2.1 Problematic silent discard?

> *Where a Packet is too large to send, the Gateway MUST discard it without sending it and then behave as if it had completed sending that Application Message.*
>
> [3.1.4.6 Max Packet Size]

If the discarded packet is part of a larger flow (like `PUBLISH QoS > 0`), the gateway should "pretend" the whole flow to the MQTT broker. Otherwise, the MQTT would consider the flow unfinished and will keep retrying. I think the spec should more explicitly state this.

Unfortunately, this also means that the transparent gateway should also implement a part of the MQTT-SN client functionality. I haven't thought through all the situations which should be taken into account, but this aspect seems troublesome to me.

2.2 MQTT Reason Codes with no MQTT-SN counterpart

I'm not sure what should transparent gateway do if it receives MQTT reason code that can't be directly translated into MQTT-SN reason code (e.g. *Topic Name Invalid*).

The MQTT-SN reason codes list should probably be extended so that every MQTT reason code has its MQTT-SN counterpart?

2.3 Free-form *Auth Method*

The draft leaves the content of the *Auth Method* entirely on the implementer. I'm afraid this could lead to an interoperability nightmare in the future.

Shouldn't the content be at least half-specified (leaving room for an implementer to add his own methods)? For instance: The Auth Methods of the form `"SASL <mechanism>"` must implement mechanisms as defined in the SASL mechanisms register (`https://www.iana.org/assignments/sasl-mechanisms/sasl-mechanisms.xhtml`) — e.g. "SASL PLAIN", "SASL OTP" etc.

Of course, the MQTT-SN specification should ideally precisely define all allowed auth Mechanisms and their implementation but I don't know if it's a viable option...

2.4 "Non-zero length alias"?

> *[...] a Topic Alias which has a value that is a non-zero length*
>
> [3.1.14.3 Reason code]

I don't understand what a "non-zero length alias" should be.

2.5 `SUBACK` and pre-defined topic alias

> *In case of "accepted" the value that will be used as topic alias by the gateway when sending `PUBLISH` packets to the client (not relevant in case of subscriptions to a short topic name or to a topic name which contains wildcard characters)*
>
> [3.1.19.3 Topic Alias]

The Topic Alias field could be zero also for `SUBSCRIBE` using a pre-defined topic alias. There's no point to return in `SUBACK` the same value that was already present in the `SUBSCRIBE` IMHO.

2.6 Missing optional values not explicitly specified

There are various fields defined as optional but there's no explicit definition of how to signalize the field is missing. In most (all?) cases it's just setting the value to zero (e.g. `PINGREQ`/*Max Messages*, `DISCONNECT`/*Session Expiry Interval*) but it should probably be explicitly mentioned.

2.7 Starting another flow where at most one is allowed

There could be e.g. only one in-flight `REGISTER` flow. What should the gateway do if a client starts another one before the previous one is finished? Should it consider this a protocol violation and close the connection? Or just ignore the second packet and let the client resend it later? I think the latter makes more sense.

2.8 Unsuccessful `REGACK` on gateway's publish

> *The client could reject the `REGISTER` packet with a `REGACK` packet indicating the rejection reason; this corresponds to an unsubscribe to the topic name indicated in the `REGISTER` packet*
>
> [4.14 Gateway's Publish Procedure]

I understand that this feature partly solves the problem of unsuccessful registration of a topic alias in a gateway's publish flow. But because it works for non-wildcard subscriptions only, there must be another mechanism to deal with failed registration on wildcard topic subscriptions. Which is not defined if I didn't overlook it. IMHO, the gateway can only ignore or retry.

Anyway, two different mechanisms for wildcard and non-wildcard subscriptions complicate the specification and gateway implementation. I think one universal behavior would be better.

2.9 *Client Id* in PINGREQ (sleep)

> *Like the CONNECT packet, this PINGREQ packet contains the Client Id*
> [4.18 Support of sleeping clients]

What is the purpose of the *Client Id* in this packet? The gateway must be able to identify the client using its network address anyway.

What should gateway do if the given *Client Id* was different than the *Client Id* given by the previous CONNECT? If it should update the session, it would also have to re-authenticate the client. And a transparent gateway should re-establish the MQTT connection. This is problematic (see 1.2).

2.10 Is CONNECT in *Asleep* or *Awake* "full-featured"?

The problem is, whether the CONNECT packet sent in *Asleep* or *Awake* is only a "signal" that a client wants to go to the *Active* state or everything that holds for a session-establishing CONNECT also holds in this situation.

If the latter is true, this would again be problematic, especially for a transparent gateway.

For a detailed description of the problem, please see
https://github.com/energomonitor/bisquitt/blob/main/doc/specification-interpretation.md

2.11 Connection closing conditions not very clear

I think the specification should explicitly state on which condition the client and gateway can/MUST close the connection. MUST (?) they resend the DISCONNECT packet until confirmed by the other party's DISCONNECT and close the connect only after that? (Or without DISCONNECT on protocol violation, unsuccessful AUTH, etc.)

[4.17 Client's Retransmission Procedure] says that "all packets" must be retransmitted, which should also apply to DISCONNECT, but I think [4.16 Client's Disconnect Procedure] should explicitly state this.

3 Thoughts, suggestions, etc.

3.1 AUTH packets ordering

The authentication flow can include multiple AUTH packets but there is no means of AUTH packets ordering. Because MQTT-SN operates over a non-reliable network, any packet can be lost or duplicated and the client and gateway have no way to detect it. So every authentication method which uses multiple AUTH packets would have to implement its own sequence number or other similar ordering/loss detection mechanism. This should probably be replaced by one universal mechanism like adding a Packet Identifier field to the AUTH packet.

3.2 Topic aliases reuse

> *Non-normative comment*
> *The gateway should attempt to make the best effort to reuse the same topic alias' mappings that existed during any initial associated ACTIVE states.*
>
> [4.18 Support of sleeping clients]

I don't see a big advantage in this if it is not obligatory. Slightly more convenient debugging perhaps? Because if it's not obligatory, it doesn't save any code/resources on the client anyway...

3.3 "Synchronous" flows are inefficient

> *At any point in time a client may have only one REGISTER packet outstanding, i.e. it has to wait for a REGACK packet before it can register another topic name.*
>
> [4.8 Topic Name and Topic Filter Registration Procedure]

This makes registration of multiple aliases synchronous hence time inefficient because every registration involves network roundtrip time and possibly retransmission timeout. If multiple parallel `REGISTER` flows were allowed, these timeouts would not sum up and I don't think this complicates the gateway or client implementation much.

The synchronization of flows is especially harmful for sleeping clients (see 1.8).

(This comment of course applies to all the flows where this rule is stated in the specification, like `PUBLISH` with QoS > 0).

3.4 Suggestion: reversed `REGISTER` logic

(This is just a suggestion and I'm aware it would be a breaking change of the protocol.)

In MQTT-SN, both a gateway and a client must keep the topic alias map. But the gateway is the "master" which assigns the alias numbers to topics. If the alias numbers were assigned by the client, the client could take advantage of knowledge about the structure of the topics which the gateway can't have.

Let's consider e.g. a device that measures pressure on hundred different pipes and periodically sends measured values to `sensor/SEN_ID/pressure/MEAS_ID` topics. When the gateway assigns aliases numbers, the device must keep at least one hundred different aliases (100*2B) in memory. If the registration logic were reversed, the device could assign aliases algorithmically:

`sensor/SEN_ID/pressure/X` $\rightarrow$ `alias=X`

The memory footprint on the gateway would be the same, but the device could avoid using any memory for topic aliases in the best case (if it does not subscribe to any topic). We can assume the client is always more resource-constrained than the gateway.

This approach where the source of the data is also the source of the alias mapping is used e.g. in the SparkPlug protocol.