End-to-End Response Time Analysis for RT-MQTT: Trajectory Approach versus Holistic Approach

Ehsan Shahri
DETI/UA/IT
University of Aveiro
Aveiro, Portugal
ehsan.shahri@ua.pt

Paulo Pedreiras
DETI/UA/IT
University of Aveiro
Aveiro, Portugal
pbrp@ua.pt

Luis Almeida
CISTER-FEUP
University of Porto
Porto, Portugal
lda@fe.up.pt

2023 IEEE 19th International Conference on Factory Communication Systems (WFCS) | 978-1-6654-6432-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/WFCS57264.2023.10144242

***Abstract*—Nowadays, custom components are increasingly being replaced by commercially available off-the-shelf hardware and standard protocols. Additionally, emerging industrial paradigms like Industry 4.0 and IoT place new demands on requirements like scalability, transparency, adaptability and efficiency. Accordingly, application layer protocols like the Message Queuing Telemetry Transport Protocol (MQTT) are becoming more and more popular in these fields, thanks to their simplicity, scalability, low resource-usage and decoupling between end nodes. However, these protocols are not deterministic, thus being unsuitable for real-time applications. Recently the authors proposed a set of extensions to the MQTT protocol, allowing applications to explicitly specify real-time requirements that are then used by a resource manager, implemented in Software Defined Networking (SDN), to create real-time channels. This paper extends the work, providing worst-case analysis using the Holistic and Trajectory approaches. The paper also includes a set of experimental results aiming to verify the correctness of both analysis and evaluate its performance in several scenarios.**

***Index Terms*—Real-Time Systems, MQTT, Fixed-priority Non-preemptive Scheduling, Trajectory Approach, Holistic Approach**

I. Introduction

Real-time computing systems are essential in industrial domains, from robotics [1], to industrial control [2] and industrial automation [3], for real-time data processing and analysis. These systems improve efficiency and ensure smooth operation, enabling systems to control various processes and make quick as well as accurate decisions. Recent trends in the industry, including Industry 4.0 and the Industrial Internet-of-Things (IIoT), have led to the integration of information technology into industrial operations in order to increase scalability, transparency, agility, flexibility, and efficiency. However, these technologies often do not prioritize timing behavior, which can conflict with the real-time requirements of industrial operations, such as quick response times and high predictability and stability. Additionally, the increased complexity and heterogeneity of data exchanges due to these technologies, including the integration of low and high data-rate flows, low and high bandwidth flows, and critical and non-critical flows, can also present challenges in resource-constrained hardware commonly found in industrial settings.

The Message Queuing Telemetry Transport (MQTT) [4] protocol is among the most popular application-layer protocols in the scope of IoT, being increasingly used in the scope of IIoT, too. It is a lightweight protocol designed to allow the interchange of small amounts of data among potentially large networks composed of simple digital devices. Its popularity stems from its simplicity, low footprint, scalability, and effective publisher-subscriber messaging model, that fit resource-constrained devices.

MQTT is normally used over TCP/IP networks, which natively provide ordered and lossless bi-directional channels, and, on top of that, offers three levels of Quality-of-Service (QoS). However, all these QoS levels are related to handling transmission errors, thus lacking entirely support for real-time requirements. Recently, the authors addressed this issue, proposing an architecture called RT-MQTT that extends MQTT with real-time services ([5], [6], [7]). This architecture enables applications to define real-time requirements over MQTT that are then translated into network reservations, enforced by Software Defined Networking (SDN) [8], more specifically using the OpenFlow protocol [9]. This paper extends the work by refining the formal system model, complementing the previous analytic approach based on the Holistic Approach (HA) with the more efficient Trajectory Approach (TA) to improve the tightness of the results, and, finally, verifying the correctness and relative performance of both analyses by means of extensive experiments.

The rest of the paper is structured as follows. Section II discusses traffic schedulability analysis for multi-hop networks. Section III introduces the RT-MQTT architecture and the corresponding system model. The response-time analyzes are presented in Section IV. Section V presents experimental results and, finally, Section VI concludes the paper and discusses future work.

This work is funded by Portuguese national funds through FCT/MCTES and, when applicable, co-funded by European Community funds, under projects IT-UIDB/50008/2020-UIDP/50008/2020 and CISTER-UIDB/04234/2020, as well as the FCT scholarship PD/BD/137388/2018. Additionally, this work is a result of project MIRAI - Machine intelligence techniques for smart and sustainable planning and operation of IoT and Edge computing applications, with reference POCI-01-0247-FEDER-069522, co-funded by the European Regional Development Fund (ERDF), through the Operational Programme for Competitiveness and Internationalization (COMPETE 2020) and the Lisbon Regional Operational Programme (LISBOA 2020), under the PORTUGAL 2020 Partnership Agreement.

II. Related Work

The scientific literature reports several methods to determine the worst-case response time in real-time multi-hop networks. The more relevant ones are briefly reviewed in this section.

Network Calculus (NC) [10] considers that network elements and arrival flows are characterized by a service curve and arrival curve, respectively. Provided with this information, NC allows computing the maximum delay that each flow can suffer at each network element, as well as the maximum size of the waiting queues and the corresponding departure curves. NC provides deterministic results, but requires the determination of arrival and service curves, usually in the form of bounds, which introduces some degree of pessimism. This pessimism can be reduced by considering the serialization effect and by grouping flows sharing common physical links [11]. A NC-based analysis has been proposed to compute the worst-case delay of Rate-Constrained (RC) traffic with the consideration of the static scheduled time-triggered (TT) frames in TTEthernet [12]–[14]. Both analyses compute the end-to-end delay of a frame by simply summing up the worst-case delay on each data-flow link traversed by the frame, which potentially leads to unreachable scenarios as the worst-case delay on several data-flow links. In a similar way, a response time analysis has been proposed [15], [16] in TTEthernet, however, it might not thoroughly cover all of the most significant scenarios. Additionally, [17] has presented a method for response time analysis that does not account for the backlog of the outgoing queue before the RC frame under analysis reaches the current node, potentially leading to an underestimation of the worst-case delay.

Another method is the Trajectory Approach (TA) [18], which computes the latest starting time of a packet on the last node visited, then moving backwards through the sequence of nodes visited by the packet (i.e. the packet trajectory), identifying the preceding packets and busy periods that affect the latency of the packet in each node. This approach is, in general, the tightest one, but is more complex to implement and validate, particularly when considering unconstrained routing schemes in which flows can cross multiple times. The TA has been optimized by considering the impact of the serialization of flows sharing the same link [19], [20], improving the tightness of the end-to-end response time estimation when compared with NC. However, there is still some inherent pessimism, such as the overestimation of competing workload, double counting frames with the maximal packet length, and the underestimation of the serialization effect. This pessimism has been analyzed in [21] and optimistic problems, pushed by the current form of optimized TA, have been shown in [22] while a solution has been proposed in [23].

The Holistic Approach (HA) [24] is one of the first methods that were developed for response time analysis. Aiming originally to distributed systems, it computes the minimum and maximum response-time of the tasks, using this jitter to compute the minimum and maximum response-times of the messages they generate. The jitter is then used to reassess the response time of the tasks triggered by the messages. The whole process is repeated until the response time of tasks and messages converge. The HA can also be applied to multi-hop switched networks. In this case, the response time and jitter of each message are computed and accumulated at each hop. The jitter accumulated at each hop is used as the release jitter for the following one, while the the response time is simply added up, thus the response times are computed in just one pass, from the source to the destination. Despite being more pessimistic than the other approaches, as it considers worst-case scenarios on every node that, often, cannot occur, HA is simple to implement and fast to execute [25].

III. RT-MQTT Network Architecture

The RT-MQTT network architecture is based on the MQTT protocol [4], which is a lightweight, scalable and flexible messaging protocol, commonly used in application domains such as Machine-to-Machine (M2M), IoT and IIoT. MQTT uses a broker-based publish-subscribe model, in which (publisher) nodes send messages to a broker that then distribute them to interested receiver nodes (subscribers). In MQTT messages are associated with a given subject, called topic, that is used to identify the set of receiver nodes. MQTT supports three QoS levels, all related with transmission error handling (QoS 0/Once, QoS 1/At Least Once and QoS 2/Only Once). It is important to notice that none of these QoS levels comprehends the specification of real-time requirements.

User properties, that appeared in MQTT V5.0, are a set of key-value pairs that can be included in messages to provide additional information about themselves and/or the associated topic. These key-value pairs are not part of the standard, being instead defined by the application. In the context of real-time systems, this field can be used to convey real-time attributes such as priority, deadline, periodicity or criticality level.

The core network elements of RT-MQTT are OpenFlow switches. The OpenFlow (OF) protocol [9] is a standard in Software-Defined Networking (SDN) [8] that defines the communication between an SDN controller and network switches. The controller decides how data packets should be forwarded through the network and these forwarding decisions are then communicated to the switches, in the form of Flow Tables. Flow tables are, in some sense, equivalent to the MAC/CAM tables of conventional switches, specifying what the OF switch should do with incoming packets.

The RT-MQTT network architecture (Figure 1) is based on the MQTT application layer protocol supported by OpenFlow switches, comprising an OpenFlow controller (OF-Controller), OpenFlow switches (OF-Switches), an MQTT broker, a real-time network manager (RT-NM) and MQTT clients (IIoT nodes). The OF-Controller is connected to the OF-Switches, having a global view of the network and storing all information in the OF-DataBase (OF-DB). RT-MQTT allows applications to explicitly specify real-time requirements, which are then conveyed in the User Properties of MQTT packets. The RT-NM intercepts all MQTT messages to extract possible real-time requirements data. It is logically placed between MQTT

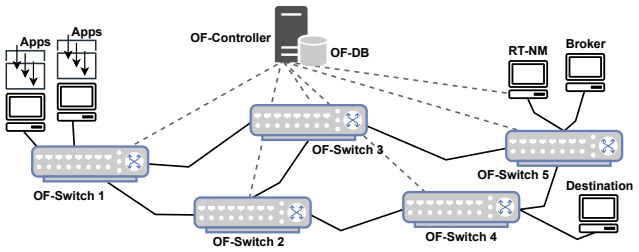


Fig. 1. High-level RT-MQTT system architecture.

clients and the broker, desirably executing in the same node. These requirements are subsequently conveyed to the OF-Controller that processes them and manages the flow tables of the OF-Switches to create corresponding real-time channels.

A. Message Model

RT-MQTT classifies packet flows (or simply messages) as `real-time` (RT) or time-sensitive (TS), and `non-real-time` (NRT). The former class refers to messages subject to strict real-time requirements, while the latter one encompasses normal MQTT messages and general background traffic. Client nodes are assumed to be supported by an operating system with real-time capabilities and can generate real-time and non-real-time traffic concurrently.

The message set Γ is composed of N sporadic messages m_i, defined as follows:

$$\Gamma = \{m_i(C_i, D_i, T_i, P_i, SRC_i, DST_i, \mathcal{L}_i, n_i) \;\; i = 1...N\} \tag{1}$$

The semantics of the parameters is the following:

C_i : message transmission time (including overheads);
D_i : deadline, defined as the maximum allowed time between the beginning of a message transmission at the source node and its reception at any of the receiver nodes;
T_i : minimum interval between consecutive message transmissions, with $D_i \leq T_i$;
P_i : message priority (ascending order, higher numeric value corresponds to higher priority);
SRC_i/DST_i : source/destination node;
$\mathcal{L}_i$: set of n_i links crossed by message m_i;
n_i : number of links that m_i crosses, i.e., $n_i = |\mathcal{L}_i|$.

As usual in IIoT applications, we consider single-packet messages. Each message crosses one or more switches, being the first one named the ingress switch and the last one the egress switch. Each element in $\mathcal{L}_i$ has a duplet $l =< x, y >$ representing a link l between node/switch x and node/switch y. A link between a node and a switch is called local-link, while a link between two switches is an inter-link. The direction of message transmission in that link is indicated by the sequence within the duplet.

Since MQTT generally relies on unicasting [26], the analysis considers only unicast streams. Scenarios with multiple subscribers are straightforward to handle, requiring the application of the analysis to each one of the corresponding broker to subscriber links.

B. Scheduling Model

Accordingly with the characteristics and capabilities of current OF-switches, it is adopted a non-preemptive fixed priority scheduling model, with FIFO ordering within each priority level. Based on the priority level of a given message m_i, the following message subsets are defined:

- $hp_i = \{j \in [1, N], P_j > P_i\}$, subset of messages having a priority strictly higher than the priority of m_i;
- $sp_i = \{j \in [1, N], j \neq i, P_j = P_i\}$, subset of messages that have the same priority as m_i, except m_i;
- $lp_i = \{j \in [1, N], P_j < P_i\}$, subset of messages having a priority strictly lower than the priority of m_i.

When a message arrives at an ingress port of a switch it is processed by the OF-pipeline and placed at a given output port queue, according to its path and priority. After being placed at an output port queue, message m_i can suffer two types of delays: i) `blocking` delay caused by one message belonging to lp_i that is already in transmission (non-preemption), and ii) `interference` delay created by the hp_i (priority scheduling) and sp_i (FIFO ordering at each priority level) message sets. Note that each output port corresponds to a link in the message path.

The delay that results from the packet processing by the OF-pipeline at each switch is modeled as the Switching Delay (SD). The SD depends on a number of factors that include specific hardware and software switch characteristics, input load, number and size of flow tables and the complexity of the action set, being usually measured experimentally to obtain a conservative bound.

The response time of a real-time message in a publisher-subscriber architecture such as RT-MQTT can be decomposed in three main parts: (i) upstream from `publisher` to `broker`, (ii) processing time at the `broker`, and (iii) downstream from `broker` to `subscriber(s)`. Parts (i) and (iii) are communication delays while (ii) is a computational delay. The response time analysis for parts (i) and (iii) is similar, only differing in the source and destination nodes. Therefore, we will only focus on part (i), specifically analyzing the response time for traffic from publishers to the broker. Several approaches on the literature address the real-time behaviour of MQTT brokers (e.g. [27], [28]), which can be combined with the contribution of this paper to compute the global publisher-subscriber delay.

Finally, it should be remarked that this paper does not consider the message set-up phase, being assumed that all resources are set and that the system configuration remains static. RT-MQTT allows dynamic updates to the real-time traffic, but the evaluation of the response-time of such updates is out of the scope of this paper.

IV. Response Time Analysis

This section presents the adaptation of the Holistic and Trajectory approaches to the RT-MQTT architecture. TA was

chosen because of its tightness, but since it is more complex to implement and validate, particularly when considering unconstrained routing schemes in which flows can cross multiple times, the simpler but more pessimistic HA-based analysis was also included. The results obtained with both approaches are experimentally verified and compared (Section V).

A. Response Time Analysis Using the Holistic Approach

According to the HA, the full path $\mathcal{L}_i$ from the source node to the destination one is analyzed link by link, being the total response time obtained by adding the response times of all individual links. The link delay computation includes interference, blocking and switch delays, and jitter is accumulated over the path. To preserve full compatibility with standard nodes, the non-real-time traffic is not declared to the RT-MQTT framework, so its characteristics are unknown. Therefore, it is assumed that the blocking delay experienced by real-time messages can be as large as the link Maximum Transmission Unit (MTU).

1) Worst-case response time: in each link (source node or OF-switch output) messages are placed in prioritized queues. Within each queue/priority level messages are handled in FIFO and transmitted non-preemptively. This model can be analyzed with the well-known response time without preemption in uniprocessors based on cumulative delays method [29].

A crucial aspect of this analysis is determining the critical instant, i.e., the message release pattern that leads to the worst-case interference that a message can suffer. In this case the critical instant occurs when a message m_i is released immediately after the release of a lower priority message with maximum size (causing maximum blocking) and immediately after all equal priority messages and together with all higher priority messages considering their maximum release jitter, assuring maximum interference from both same and higher priority messages.

Non-preemptive transmissions are subject to the push-through effect according to which a given instance of a message m_i can delay higher priority messages through blocking that in turn will generate higher interference in following instances of m_i itself. For this reason, the worst-case response time of m_i in a given output port may occur at instances beyond the one that is released at a critical instant, within the so-called **occupied period**. The number of instances following a critical instant that have to be checked to determine the worst-case response time is given by $Q_i = \lceil (w_i + J_i)/T_i \rceil$, where J_i is the release jitter and w_i is the length of the level-i busy period. This period is computed as in Equation 2 using fixed point iteration, starting with $w_i^0 = B_i + C_i$ and ending when $w_i^{n+1} = w_i^n$, where B_i is the blocking delay caused by NRT and lp_i messages.

$$w_i^{n+1} = B_i + \sum_{j \in hp_i \cup sp_i} \left\lceil \frac{w_i^n + J_j}{T_j} \right\rceil .C_j \qquad (2)$$

The level-i occupied period starts at the critical instant and extends until the following level-i **idle period**, which is the instant when the queues of priorities P_i and above of the output port become empty. The worst-case delay that each one of the Q_i messages instances that fall inside the level-i occupied period can experience before starting transmission must be computed individually, including for each one the load due to the q previous instances. Equation 3 computes an upper bound to these delays, expressed as $v_i(q)$, and can also be solved through fixed-point iteration with a possible initial value $v_i^0(q) = B_i + qC_i$ and ending when either $v_i^{n+1}(q) = v_i^n(q)$ or when $v_i^{n+1}(q) + C_i - qT_i > D_i - J_i$ in which case the deadline cannot be guaranteed.

$$v_i^{n+1}(q) = B_i + qC_i + \sum_{j \in hp_i \cup sp_i} \left(\left\lfloor \frac{v_i^n(q) + J_j}{T_j} \right\rfloor + 1 \right) .C_j \qquad (3)$$

The worst-case response time of an instance preceded by q instances can then be obtained with $RT_i(q) = v_i(q) + C_i - qT_i$. Equation 4 gives us the worst-case response time for m_i.

$$RT_i = max_{q=0,1...Q_i-1}(v_i(q) + C_i - qT_i) \qquad (4)$$

2) Response time calculation algorithm: Algorithm 1 shows the computation of the worst-case response time of a message m_i, from its source node until the destination one, taking as input the network topology and the message set.

Algorithm 1: WCRT calculation for m_i using HA.

Input: Γ
Output: RT_i^{Total}

```
/* A. Compute delays and jitter at source node:      */
RT_i^Total = ResponseTimeCalc(i, 1)
J_i^acc = J_i + J_{i,1}^QP
k = 2
/* B. Compute delays and jitter for each switch:     */
while k ≤ n_i do
    SD_{i,k} = SwitchingDelayCalc(i, k)
    J_i^acc = J_i^acc + J_{i,k}^SD
    RT_{i,k} = ResponseTimeCalc(i, k)
    J_i^acc = J_i^acc + J_{i,k}^QP
    RT_i^Total = RT_i^Total + RT_{i,k} + SD_{i,k}
    k = k + 1
end while
```

The algorithm has two parts. The first one (A) processes the output link of the source node (i.e. k=1), including the computation of the response time ($ResponseTimeCalc(i, 1)$ according to Equation 4) and output jitter (J_i plus $J_{i,1}^{QP}$, which is the response time jitter). The response time jitter is calculated by subtracting the best-case response time (the message's fastest possible transmission time) from the worst-case response time. These values are used to initialize two accumulators that shall account for the response time (RT_i^{Total}) and release jitter (J_i^{acc}) for the remaining links along the path.

The second part of the algorithm (B) is very similar to the first one. The only difference is that it also considers the bounded switching delay introduced by the switch under analysis and the additional jitter it may cause. This is repeated from the second to the last link in the path of m_i. Please

note that the switching delay is measured experimentally, as discussed in Section III-B.

B. Response Time Analysis Using Trajectory Approach

The TA identifies the set of messages that postpone a given message m_i generated at time t on its path $\mathcal{L}_i$ from the source to the destination node, evaluating its impact in the response time on the visited links, starting from the last one backward until the source link. On each link, the identified messages create a period called the busy period. The sum of these periods and the total transmission delay of the network allows computing the response time of m_i.

1) Worst-case response time: as we consider fixed-priority non-preemptive scheduling, the processing of a message can no longer be delayed after it has started. Therefore, the TA [18], [30] recursively computes the latest starting time of m_i on its last visited switch $last_i$ (Equation. 5). Notice that $(1 + \lfloor x \rfloor)^+$ stands for $\max(0; \lfloor x \rfloor)$. Moreover, $first_{j,i}$ (resp. $last_{j,i}$) is the first switch (resp. last switch) visited by m_j on path $\mathcal{L}_i$, $slow_{j,i}$ indicates the slowest switch visited by m_j on path $\mathcal{L}_i$ and $pre_i(sw)$ is the switch visited before switch sw by m_i. In turn, $Smax_i^{sw}$ denotes the maximum time taken by m_i to go from its source switch to switch sw. Finally, $Smin_j^{sw}$ (resp. $Smax_j^{sw}$) indicates the minimum (resp. maximum) time taken by m_j to go from its source switch to switch sw and $A_{j,i} = Smax_j^{first_{j,i}} - M_i^{first_{j,i}} + J_j$.

The notation of $M_i^{first_{j,i}}$ represents the minimum time required by a message to travel from the source of m_i to switch $first_{j,i}$.

$$W_{i,t}^{last_i} = \sum_{j \in hp_i} \left(1 + \left\lfloor \frac{W_{i,t}^{last_{j,i}} - Smin_j^{last_{j,i}} + A_{j,i}}{T_j} \right\rfloor\right)^+ .SD_j^{slow_{j,i}} + \sum_{j \in sp_i} \left(1 + \left\lfloor \frac{t + Smax_i^{first_{j,i}} - Smin_j^{first_{j,i}} + A_{j,i}}{T_j} \right\rfloor\right)^+ .SD_j^{slow_{j,i}} + \sum_{\substack{sw \in \mathcal{L}_i \\ sw \neq slow_i}} \max_{j \in hp_i \cup sp_i} \{SD_j^{sw}\} + (|\mathcal{L}_i|.C_i) + B_i - SD_i^{last_i} \tag{5}$$

$$M_i^{first_{j,i}} = \sum_{sw=first_i}^{pre_i(first_{j,i})} (\min_{j \in hp_i \cup sp_i} \{SD_j^{sw}\} + C_i) \tag{6}$$

The first term of $W_{i,t}^{last_i}$ corresponds to the maximum delay incurred by m_i due to packets having higher fixed priorities. The second term represents the maximum delay incurred by m_i due to messages having the same fixed priorities. The third term is the switching delay of the messages with higher or same fixed priorities for each switch of path $\mathcal{L}_i$, except the slowest one. The fourth term indicates the maximum transmission delay on links that m_i crosses in the path $\mathcal{L}_i$. B_i represents the delay directly due to the non-preemptive effect (blocking), being given by Equation 7.

$$B_i = \left(\max_{j \in lp_i} \{SD_j^{first_i}\} - 1\right)^+ + \sum_{\substack{sw \in \mathcal{L}_i \\ sw \neq first_i}} \left((\max_{\substack{j \in lp_i \\ sw = first_{j,i}}} \{SD_j^{sw}\} - 1); (\max_{\substack{j \in lp_i \\ sw \neq first_{j,i}}} \{SD_j^{sw}\} - SD_i^{sw-1}) \right)^+ \tag{7}$$

Finally, the last term is subtracted since $W_{i,t}^{last_i}$ is the start time of message m_i transmission at the output port $last_i$. Consequently, Equation. 8 gives us the worst-case response time for m_i using TA.

$$RT_i = W_{i,t}^{last_i} + SD_i^{last_i} - t \tag{8}$$

2) Response time calculation algorithm: algorithm 2 shows the computation of the worst-case response time for m_i for the total route from source to broker using TA, taking as input the network topology and the message set.

Algorithm 2: WCRT calculation for m_i using TA.

Input: Γ
Output: RT_i^{Total}

/* A. Initialize the iteration: */
$sw = 1$
$Smax_j^{first_{j,i}}(1) = \sum_{sw=first_j}^{pre(first_{j,i})} (SD_j^{sw} + C_i)$
/* B. Compute the latest starting time and response time: */
while *true* **do**
  $sw = sw + 1$
  for $m_j \in (hp_i \cup sp_i)$ **do**
    for $sw \in \mathcal{L}_i$ **do**
      if $(sw = last_i)$ *or* $(m_j$ *cross* m_i *such that* $sw = last_{j,i}$ *or* $sw = pre_i(first_{j,i})$ **then**
        $W_{i,t}^{sw} = LatestStartTimeCalc(sw, i)$
        if $(m_j$ *cross* m_i *such that* $sw = pre_i(first_{j,i}))$ **then**
          $Smax_i^{first_{j,i}}(sw + 1) = W_i^{sw}(t) + SD_i^{sw} - t + C_i$
          if $Smax_i^{first_{j,i}}(sw + 1) = Smax_i^{first_{j,i}}(sw)$ **then**
            break
        if $sw = last_i$ **then**
          $RT_i^{Total} = ResponseTimeCalc(i)$
          if $RT_i^{Total} > D_i$ **then**
            break
      end if
    end for
  end for
end while

The algorithm has two parts. The first part (A) calculates the $Smax_j^{first_{j,i}}(sw)$ (corresponding to $sw = 1$) for any message $m_j \in (hp_i \cup sp_i)$ that cross m_i. This value is used to initialize the iteration that will allow computing the response time for the total route (RT_i^{Total}). In the second part (B), the algorithm starts to compute the worst-case response time for m_i, iteratively. The algorithm considers any message $m_j \in (hp_i \cup sp_i)$ to find the worst-case scenarios incurred by m_i. It computes the latest starting time ($LatestStartTimeCalc(sw, i)$ according to Equation 5) in the last switch visited and calculates then the worst-case response time ($ResponseTimeCalc(i)$ according to Equation 8) for m_i. The algorithm is stopped as soon as there exists a message such that its end-to-end response time exceeds its end-to-end deadline.

V. Performance Assessment

This section presents an empirical study using the Mininet emulation framework in multiple scenarios of varying complexity and load levels, with the objective of verifying the correctness and relative performance of the analyses. For consistency, for all cases it is measured the upstream path delay, between the publisher's side (writing to the respective socket) and the corresponding reception at the broker. This experimental value is then compared with the analytical response times computed both with HA and TA.

A. Emulation Setup

For obtaining the experimental results it was used the Mininet virtual network emulator, version 2.3.0d6 http://mininet.org/, together with Eclipse Mosquitto [31] (v2.0.10) and Eclipse Paho MQTT library to create the MQTT clients and broker. Mininet is executed on a laptop computer featuring a 4.9 GHz Intel Core i7 processor and 16 GB of RAM. The SDN controller is the RYU OF-Controller https://ryu-sdn.org/ and it is executed on the same laptop computer.

The impact of retransmissions on the real-time traffic response time is out of the scope of this paper and is not accounted for in the analysis. Nevertheless, to approach realistic scenarios, in the experiments the QoS of all MQTT messages is set to 1 (deliver at least once) and the error rate was set to 3%. Therefore, the experimental workload is higher than the one considered in the analysis. Since the analysis are pessimistic and the error rate is low, the measured response-times were still below the analytic bounds, despite this unfavorable scenario, as can be seen further on.

The experiments consider the network topology shown in Figure 2. The topology includes the OF-controller (`c0`), 10 OF-Switches (`s1` to `s10`), 53 MQTT publishers (nodes `h1` to `h48` and `h52` to `h56`), each publishing periodically a message (m_1 to m_{48} and m_{52} to m_{56}, resp.) to a unique topic. The broker and the `RT-NM` are both hosted at node `h49`. All MQTT topics are subscribed by a single subscriber placed at node `h50`, including both normal MQTT topics (i.e. NRT traffic) from nodes {`h8, h16, h24, h32, h40, h48, h52 ...h56`}, and time-sensitive MQTT topics from the remaining nodes. The NRT traffic also includes supplementary sources such as the *Distributed Internet Traffic Generator* (D-ITG) http://traffic.comics.unina.it/software/ITG/ for generating TCP packets from node `h16`, *VLC media player* for generating audio/video streams from node `h24`, and *vsftpd* for transferring files via the File Transfer Protocol (FTP) https://linuxconfig.org/how-to-setup-and-use-ftp-server-in-ubuntu-linux from node `h32`, all directed to node `h51`, aiming to mimic the diversity of industrial scenarios using heterogeneous data exchanges. The bandwidth for D-ITG, VLC, vsftpd, and normal MQTT sources is limited to 38 Mbit/s, 1.2 Mbit/s, 3.2 Mbit/s and 6 Mbit/s, respectively. The link speed is set to 100 Mbit/s, which is a common setting in industrial applications.

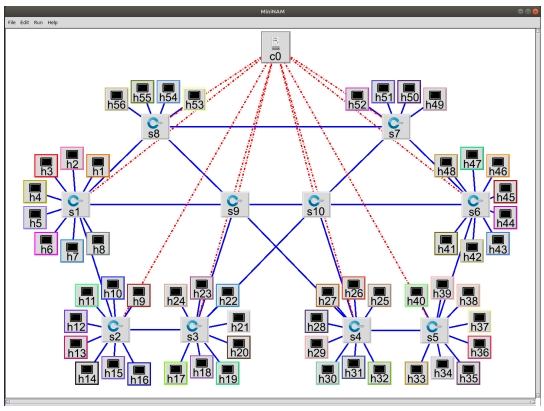


Fig. 2. Network topology used in MiniNet.

RT-MQTT messages are evenly assigned to 7 different classes, each with a given deadline and priority, according to the Deadline Monotonic policy, as shown in Table I. The distribution of messages through the distinct classes places nodes with the same priority in different switches and nodes with different priorities in the same node. This distribution was designed to allow separating the effect of paths and priorities on the WCRT. MQTT publications are not synchronized, following a nominal period in the interval $[10\ 20] \pm [0\ 0.4]$ ms, generating varying interference patterns. These publications use a single Ethernet packet with a maximum size (1500 bytes), leading to a transmission time of $123\mu s$.

The evaluation is carried out with three distinct load levels, measured at the broker link, labeled `Low`, `Medium`, and `High`. These configurations use 12, 24 and 42 RT MQTT publishers, with a maximum bandwidth utilization of 9.1 Mbit/s, 18.2 Mbit/s, and 31.9 Mbit/s, respectively. For the `Low` load level only two RT publishers are connected to each switch $s1$ to $s6$. For the `Medium` load level, two additional RT publishers are added to each switch and, finally, for the `High` load level, all RT publishers are active. Table I reports the active RT messages for each load level.

The path traversed by each message is as follows:

$\mathcal{L}_1 = \{< hA, s1 >, < s1, s8 >, < s8, s7 >, < s7, h49 >\}$;
$\mathcal{L}_2 = \{< hB, s2 >, < s2, s3 >, < s3, s10 >, < s10, s7 >, < s7, h49 >\}$;
$\mathcal{L}_3 = \{< hC, s3 >, < s3, s10 >, < s10, s7 >, < s7, h49 >\}$;
$\mathcal{L}_4 = \{< hD, s4 >, < s4, s10 >, < s10, s7 >, < s7, h49 >\}$;

TABLE I
MESSAGE SPECIFICATION.

Message Set	Load Level	Origin Switch						P_i	D_i
		$s1$	$s2$	$s3$	$s4$	$s5$	$s6$		
		Crossed Path							
		$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_4$	$\mathcal{L}_5$	$\mathcal{L}_6$		
	High	m_1	m_9	m_{17}	m_{25}	m_{33}	m_{41}	7	10
	Medium	m_2	m_{10}	m_{18}	m_{26}	m_{34}	m_{42}	6	12
	Low	m_3	m_{11}	m_{19}	m_{27}	m_{35}	m_{43}	5	14
		m_4	m_{12}	m_{20}	m_{28}	m_{36}	m_{44}	4	16
		m_5	m_{13}	m_{21}	m_{29}	m_{37}	m_{45}	3	18
		m_6	m_{14}	m_{22}	m_{30}	m_{38}	m_{46}	2	19
		m_7	m_{15}	m_{23}	m_{31}	m_{39}	m_{47}	1	20

$\mathcal{L}_5 = \{< hE, s5 >, < s5, s6 >, < s6, s7 >, < s7, h49 >\}$;
$\mathcal{L}_6 = \{< hF, s6 >, < s6, s7 >, < s7, h49 >\}$.

where hA to hF indicate the local-link connections to the ingress switches as follows:

- $hA : h1\,to\,h7$
- $hB : h9\,to\,h15$
- $hC : h17\,to\,h23$
- $hD : h25\,to\,h31$
- $hE : h33\,to\,h39$
- $hF : h41\,to\,h47$

The same configuration is used for the emulation and analysis, to ensure that the results are comparable. The `libpcap` library is used to measure jitters and switching delays by timestamping packets. Platform-induced perturbations are minimized by temporarily storing timestamps in memory until the completion of each one of the simulation runs.

B. Experimental Results

The topology was evaluated for the three load levels {Low, Medium, High} reported above, with each level being run 1000 times and publishers sending at least 100 messages in each run. For each run, experimental data was collected and analyzed, to determine the experimental WCRT of all messages. The WCRT of each of the configurations is also computed, using both HA and TA (Algorithms 1 and 2).

Figure 3 shows that the WCRT values computed using TA and HA are higher than the WCRT measured experimentally, as expected. Moreover, it is also possible to observe that TA (RT_i^{TA}) approaches more closely the experimental values (RT_i^{Exp}) than HA (RT_i^{HA}). For the former case the deviation is at most $+6.3\%$, $+8.1\%$ and $+10.7\%$ for the Low, Medium and High load levels, respectively, while for latter one the corresponding deviation is at most $+33.3\%$, $+36.1\%$ and $+38.0\%$. It can also be observed that the degree of pessimism of HA with respect to TA increases when the path length grows or the priority decreases. This observation is also in line with the expectations, as the HA tends to produce more pessimistic results by allowing infeasible interference scenarios (e.g. interference from the same message instances are accounted for in all hops) that are more pronounced when the priority is lower and/or the path longer. For example, Figure 3 c) shows that the WCRT of messages $m1$ to $m7$, which are all sent through the same path, increases with the priority, and that the pessimism of the analysis grows, particularly for the HA. Furthermore, we can observe that messages with the same priority crossing path $\mathcal{L}_2$ exhibit significantly higher response time than the ones crossing path $\mathcal{L}_6$ (e.g. $m9$/$m41$, $m10$/$m42$, ..., $m15$/$m47$), and the pessimism of the analysis is also higher for $\mathcal{L}_2$ compared $\mathcal{L}_6$, as path $\mathcal{L}_2$ has two more hops than path $\mathcal{L}_6$. On the other hand, paths with similar length (i.e. $\mathcal{L}_1$, $\mathcal{L}_3$, $\mathcal{L}_4$, and $\mathcal{L}_5$) have similar results, with the differences explained by the different load in the interlinks.

In summary, the results show TA approach is tighter than HA approach, and that the degree of pessimism is higher for lower-priority messages, longer paths, and higher load, as expected. Results confirm RT-MQTT framework enables real-time traffic transmission and can determine upper bounds for network-level latency, thus showing that RT-MQTT is predictable and suitable for real-time apps if end-nodes and broker software behave deterministically.

VI. CONCLUSIONS AND FUTURE WORK

MQTT is gaining popularity in IoT and IIoT applications, but its scope of usage is constrained by its lack of support for timeliness requirements. To overcome this, the authors proposed a framework, known as RT-MQTT, that enables reserving real-time channels over SDN/Openflow networks.

This paper presents a comprehensive study of the application of response time analysis to RT-MQTT on multi-hop SDN/OpenFlow switched networks, with focus on the Trajectory Approach for non-preemptive fixed-priority scheduling of sporadic messages, comparing its results with those obtained using the Holistic Approach. The analytic results are verified through extensive empirical testing within the Mininet emulator framework, being determined that the RT-MQTT framework is predictable and analyzable, that both TA and HA provide safe upper-bounds to network-level latency and that the TA presents a lower level of pessimism, compared to the HA, at expenses of an higher complexity. Future work includes the analysis of the broker temporal behavior to support an end-to-end (publisher-to-subscriber) delay model.

REFERENCES

[1] S. M. Kasaei *et al.*, "A reliable model-based walking engine with push recovery capability," in *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2017, pp. 122–127.

[2] J. O. Trierweiler, "Real-time optimization of industrial processes," in *Encyclopedia of Systems and Control*. Springer, 2021, pp. 1827–1836.

[3] A. Massaro *et al.*, "Systems for an intelligent application of automated processes in industry: a case study from "pmi iot industry 4.0" project," in *2020 IEEE International Workshop on Metrology for Industry 4.0 IoT*, 2020, pp. 21–26.

[4] O. Standard, "Mqtt version 5.0," *Retrieved June*, vol. 22, p. 2020, 2019.

[5] E. Shahri *et al.*, "Enhancing mqtt with real-time and reliable communication services," in *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*. IEEE, 2021, pp. 1–6.

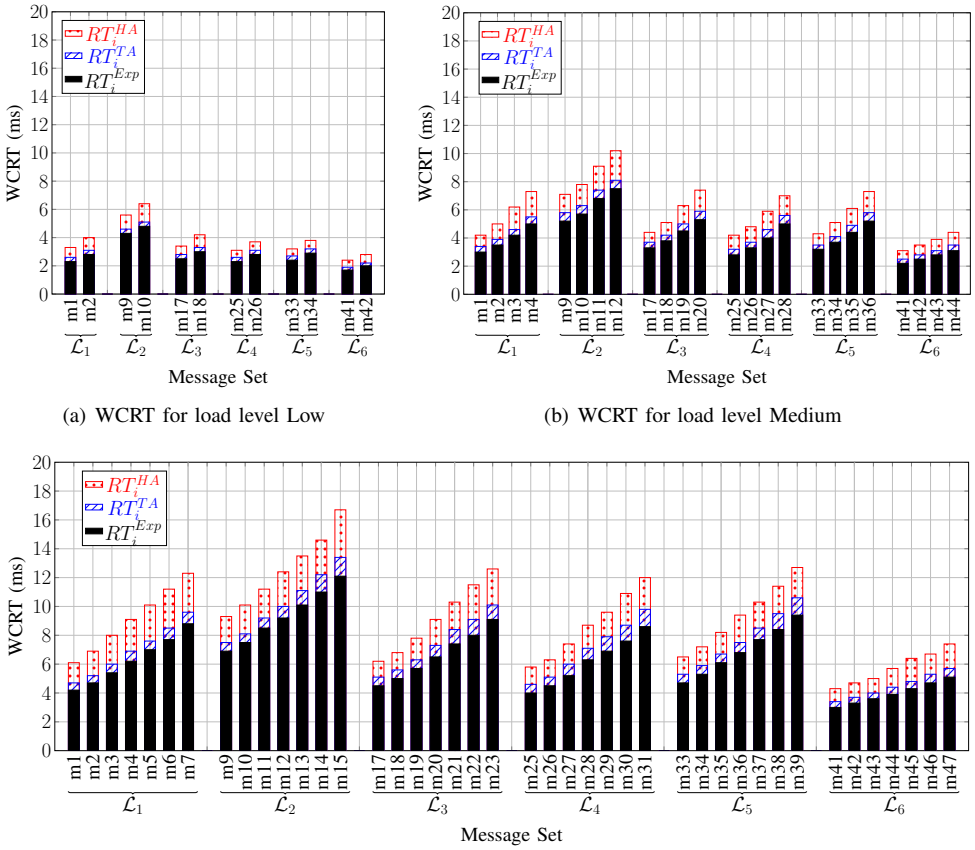
[6] ——, "Extending mqtt with real-time communication services based on sdn," in *2022 Sensor Applications in Industrial Automation (ISSN 1424-8220)*. Sensors SAIA SI, 2022, pp. 1–6.

[7] ——, "Response time analysis for rt-mqtt protocol grounded on sdn," in *EMSIG Workshop on Next Generation Real-Time Embedded Systems (NGRES)*, Tolouse, France, January 2023. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2023/17736

[8] W. Xia *et al.*, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2014.

[9] "Openflow. available online: https://opennetworking.org/sdn-resources/customer-case-studies/openflow/."

[10] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer, 2001.



(a) WCRT for load level Low

(b) WCRT for load level Medium

(c) WCRT for load level High

Fig. 3. Analytical RT_i^{HA} and RT_i^{TA} versus observed RT_i^{Exp} WCRT for the real-time messages in three load levels Low, Medium and High.

[11] Q. Xu and X. Yang, "Performance analysis on transmission estimation for avionics real-time system using optimized network calculus," *International Journal of Aeronautical and Space Sciences*, vol. 20, no. 2, pp. 506–517, 2019.

[12] W. Steiner *et al.*, "Time-triggered ethernet," in *Time-Triggered Communication*. CRC Press, 2018, pp. 209–248.

[13] L. Zhao *et al.*, "Improving worst-case latency analysis for rate-constrained traffic in the time-triggered ethernet network," *IEEE Communications Letters*, vol. 18, no. 11, pp. 1927–1930, 2014.

[14] ——, "Timing analysis of rate-constrained traffic in TTEthernet using network calculus," *Real-Time Systems*, vol. 53, no. 2, pp. 254–287, Mar. 2017. [Online]. Available: http://link.springer.com/10.1007/s11241-016-9265-0

[15] O. Kermia, "Timing analysis of ttethernet traffic," *Journal of Circuits, Systems and Computers*, vol. 24, no. 09, p. 1550140, 2015.

[16] ——, "Schedulability analysis and efficient scheduling of rate constrained messages in the ttethernet protocol," *Software: Practice and Experience*, vol. 47, no. 11, pp. 1485–1499, 2017.

[17] D. TamasSelicean *et al.*, "Timing analysis of rate constrained traffic for the ttethernet communication protocol," in *2015 IEEE 18th International Symposium on Real-Time Distributed Computing*. IEEE, 2015, pp. 119–126.

[18] S. Martin and P. Minet, "Worst case end-to-end response times of flows scheduled with fp/fifo," in *International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*. IEEE, 2006, pp. 54–54.

[19] H. Bauer *et al.*, "Improving the worst-case delay analysis of an afdx network using an optimized trajectory approach," *IEEE Transactions on Industrial informatics*, vol. 6, no. 4, pp. 521–533, 2010.

[20] ——, "Applying trajectory approach with static priority queuing for improving the use of available afdx resources," *Real-time systems*, vol. 48, no. 1, pp. 101–133, 2012.

[21] X. Li *et al.*, "Analysis of the pessimism of the trajectory approach for upper bounding end-to-end delay of sporadic flows sharing a switched ethernet network." in *RTNS*. Citeseer, 2011, pp. 149–158.

[22] G. Kemayo *et al.*, "Optimistic problems in the trajectory approach in fifo context," in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2013, pp. 1–8.

[23] D. Tămaș-Selicean *et al.*, "Design optimization of ttethernet-based distributed real-time systems," *Real-Time Systems*, vol. 51, no. 1, pp. 1–35, 2015.

[24] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocessing and microprogramming*, vol. 40, no. 2-3, pp. 117–134, 1994.

[25] J. J. Gutiérrez *et al.*, "Holistic schedulability analysis for multipacket messages in afdx networks," *Real-Time Systems*, vol. 50, no. 2, pp. 230–269, 2014.

[26] J.-H. Park *et al.*, "Dm-mqtt: An efficient mqtt based on sdn multicast for massive iot communications," *Sensors*, vol. 18, no. 9, p. 3071, 2018.

[27] C. O. Seongjin Kim, "A Study on Method for Message Processing by Priority in MQTT Broker," *JKIICE-Journal of the Korea Institute of Information and Communication Engineering*, Jul. 2017.

[28] Y.-S. K. et al, "MQTT Broker with Priority Support for Emerg. Events in IoT," *Sensors and Materials*, 2018.

[29] R. I. Davis and A. Burns, "Response time upper bounds for fixed priority real-time systems," in *2008 Real-Time Systems Symposium*. IEEE, 2008, pp. 407–418.

[30] S. Martin *et al.*, "The trajectory approach for the end-to-end response times with non-preemptive fp/edf," in *International Conference on Software Engineering Research and Applications*. Springer, 2006, pp. 229–247.

[31] R. A. Light, "Mosquitto: server and client implementation of the mqtt protocol," *Journal of Open Source Software*, vol. 2, no. 13, 2017.