Adaptive Seamless Redundancy to Achieve Highly Dependable MQTT Communication

Claudio Zunino, *Member, IEEE*, Gianluca Cena, *Senior Member, IEEE*, Stefano Scanzio, *Senior Member, IEEE*, and Adriano Valenzano, *Senior Member, IEEE*

***Abstract*—Nowadays, large enterprises, critical infrastructures, and utilities are complex systems, and consist of a number of subsystems, often located far away from each other, that coordinate their operations over geographic networks. Typically, they are interconnected through the Internet and communicate by means of transport control protocol (TCP)/IP-based protocols. When system-level resilience is required, extremely high dependability is demanded from the network. In this article a solution is described and evaluated that loosely resembles SDN architectures. A redundant version of MQTT is used for the data plane, whereas an adaptive mechanism is implemented in the control plane that evaluates the quality of paths and dynamically selects the best choice. To maximize dependability, yet keeping resource consumption to acceptable levels, concepts borrowed from reinforcement learning are exploited. An experimental campaign corroborated our expectations and showed the practical feasibility of our proposal.**

***Index Terms*—Industrial Internet of Things (IIoT), MQTT, reinforcement learning, resilience, seamless redundancy.**

I. Introduction

DEPENDABILITY has traditionally been a crucial requirement of most industrial applications. According to Laprie et al. [1], it encompasses attributes like availability, reliability, safety, integrity, maintainability, and sometimes confidentiality. Consequently, a number of solutions, practical guidelines, and countermeasures have been developed over the past decades that proved to be quite effective in making their behavior more dependable. Recent worldwide events (terrorist attacks, pandemics, wars, to which natural and human-induced disasters have to be added) have drawn more and more attention on *resilience*. When applied to large networked systems that are part of complex (critical) infrastructures, it can be defined as "the persistence of dependability when facing changes" [2]. In Layman's terms, it reflects the overall ability of such systems, on which our lives, countries, and economy often depend, to adapt to difficult situations and to keep providing an acceptable service in spite of adverse events that can be hardly foreseen and prevented.

Manuscript received 26 October 2022; revised 22 March 2023; accepted 19 April 2023. Date of publication 1 May 2023; date of current version 11 December 2023. Paper no. TII-22-4443. *(Corresponding author: Claudio Zunino.)*

The authors are with the National Research Council of Italy (CNR–IEIIT), I-10129 Torino, Italy (e-mail: claudio.zunino@cnr.it; gianluca.cena@cnr.it; stefano.scanzio@cnr.it; adriano.valenzano@polito.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2023.3271708.

Digital Object Identifier 10.1109/TII.2023.3271708

Due to the current shift of industrial automation (process and factory) to the Industry 4.0 paradigm, the related technologies, including the Industrial Internet of Things (IIoT), are gaining momentum, both in-premises (e.g., inside a plant) and worldwide (the cloud is making full integration possible and permits, e.g., large-scale global optimization). It is worth remarking that very similar trends are affecting other relevant scenarios, like utilities, smart cities, and smart agriculture, to cite a few. Communication protocols like the Open Platform Communications – Unified Architecture (OPC UA) [3] are being envisaged both on the small scale, when controllers have to be connected in real time to nearby controlled devices [4], [5], and on the large scale, where communications take place on metropolitan or geographic areas [6]. This is the case of, e.g., power production, transmission, and distribution in smart grids, which require coordination among a large number of entities (power plants, substations, wind/solar farms, large industrial users, public transportation, etc.) that are spread over wide areas, often spanning across several countries. In such contexts timings are sometimes important [7], even though relying on a large public network for communication means that no strict deadlines can be ensured for data delivery.

Another protocol that is relevant to these scenarios is MQTT, formerly known as message queue telemetry transport [8]. This solution has been conceived explicitly for the Internet of Things [9], and has become quite popular in the past years as it enables flexible machine to machine (M2M) communication and supports kind of multicasting over the Internet. To this purpose messages are labeled with *topics*, which are used to distinguish among different data flows. MQTT can be adopted as data transport for OPC UA according to the Publish–Subscribe (PubSub) mechanism, as foreseen by Part 14 of the OPC UA specification. In particular, intelligent gateways can be used as edge devices that act as a bridge between on-premises local networks and the cloud. Southbound communication typically exploits industry solutions like OPC UA and MODBUS/transport control protocol (TCP), whereas northbound communication may profitably rely on MQTT and AMQP [10] to publish data to cloud services [11].

Distributed applications communicating over the Internet cannot be more dependable than the Internet itself. This is because end users are not given any control on inner network operations. TCP proves to be reliable enough for most kinds of applications, including multimedia streaming and VoIP, which demand low latency. However, it could be unable to meet the tight timing constraints of mission-critical control applications, where the likelihood that some data may be lost or excessively delayed must be virtually close to zero. In fact, chances are that unexpected phenomena (e.g., an earthquake) may damage a portion of the Internet severely. The ability of routing protocols to find alternative paths in short times (fraction of a second up to minutes) is usually adequate for noncritical applications, but this is untrue for critical infrastructures, especially in the cases when network failures are provoked deliberately (physical and cyber attacks).

In these scenarios, a straightforward solution is provided by seamless redundancy. Such a technique was exploited about ten years ago to make industrial Ethernet fault-tolerant, as specified by the parallel redundancy protocol (PRP) [12], and later proposed for Wi-Fi [13] and CANbus [14] as well. To the best of our knowledge, the iPRP proposal in [15] is the first attempt to apply seamless redundancy to Internet connections. In that article, UDP packets were duplicated on the transmitter side and redundant copies removed on the receiver site (deduplication). While very appealing, iPRP suffered from two limitations. First, since UDP is implemented in kernel space, existing applications cannot be simply recompiled to work with iPRP. Second, only limited hints were provided in [15] on the way to make the Internet paths used by redundant connections disjoint. In particular, a suitable selection of the exit interfaces was suggested (using fields like the internet protocol version 6 (IPv6) flow label and source routing header extensions hardly works, as they are typically ignored/rejected by routers). However, little can be done by users so that the portions of the paths located in the Internet core are separate. Having a nonnegligible portion of paths in common lowers the benefits of redundancy, since any failures that affect the related routers and links provoke (usually short) gaps in data communication, which last until routing tables are reconfigured, hence restoring connection.

In this article we propose a similar approach, which is layered on top of MQTT. The advantages are twofold. First, MQTT libraries (in user space) can be modified by embedding seamless redundancy without changing their application programming interface (API), so that existing programs only need to be recompiled. Second, path diversity is now under (partial) control of the end-users, who, besides relying on distinct interfaces to the Internet, are enabled to explicitly select brokers in such a way to minimize overlap between paths. For instance, if brokers are located in separate regions/countries, a nonnegligible portion of the paths traversed by the different copies of any given message will likely differ.

Unlike the preliminary proposal in [16], in which path selection was performed statically, an architecture is exploited here where the brokers used for redundant MQTT transmission are selected automatically from a pool of available brokers distributed worldwide (e.g., hosted on virtual machines in the cloud). The selection strategy adaptively chooses the different paths that make up any redundant MQTT connection in such a way to optimize some performance indicator about communication quality, for example by trying to achieve shorter latency and less message losses.

The rest of this article is organized as follow: in Section II our solutions that apply static and adaptive seamless redundancy to MQTT are introduced, while in Section III the setup we used for experimental evaluation is described. Section IV reports on the results we obtained. Finally, Section V concludes this article.

II. Resilient Communication Over MQTT

As a matter of fact, the Internet is nowadays very reliable. Thanks to the error, congestion, and flow control mechanisms, failures to deliver information correctly when using TCP-based protocols like MQTT are quite rare in ordinary conditions. The probability that a message is lost can be lowered further by exploiting MQTT QoS 1 or 2, which ensure "at least once" and "exactly once" delivery. However, these mechanisms can do nothing in the case some intermediate link or router breaks down or suffers from quasi-permanent congestion (due to, e.g., a security attack). In these cases, resuming correct communication is up to the routing mechanisms in the Internet, and not the transport layer in end nodes.

Exploiting retransmissions improves reliability but increases latency contextually, which is hardly acceptable in systems with real-time constraints. For similar reasons, although relying on path redundancy, Multipath TCP (MPTCP) [17] does not ensure timely delivery in the case of faults that affect the Internet core. Our proposal aims at ultradependable geographic connectivity where, besides reliability and data integrity, timely delivery within the intended deadlines is also sought for the majority of the messages (users should be enabled to select the target deadline miss ratio according to their needs). To this purpose, it applies PRP seamless redundancy to MQTT communication. There are, however, clear differences. In fact, PRP demands that the underlying network equipment (made up of Ethernet switches and links) is fully duplicated, which is unfeasible for the Internet.

The solution we devised relies on some principles that are found also at the basis of software-defined networking (SDN), in particular the separation between *data plane* and *control plane*. For the former we defined a *redundant MQTT* (RMQTT) protocol, which extends standard MQTT and implements duplication and deduplication at the end points of every redundant connection, while the latter implements an adaptive mechanism that achieves an effective tradeoff between resource usage and communication dependability.

A. Redundant MQTT

The primary goal of RMQTT is to take every piece of data generated by applications on publishers and issue a number of contextual publishing requests targeted at different brokers. Every transmitted messages bears an identical copy of the original message. On subscribers, the reverse operation is performed. Each such devices subscribes to the above brokers, hence it receives multiple copies of the same message. It is up to RMQTT to discard all copies but the one that arrived first.

Mechanisms to do so efficiently are available from PRP implementations. They require that every copy is properly identified.

To this purpose, the topic in the original MQTT message is augmented by the publisher so as to include the following.

1) A sequence number (SeqNo), modeled as a counter that monotonically increases on every new message the publisher transmits.
2) The publisher ID (PubID), which uniquely identifies the publisher in those cases when more than one is using the same original topic.
3) The path ID (PathID), which identifies the broker the message will be sent to, among the pool of brokers available to every redundant connection.
4) The interface ID (IfcID), which identifies the network interface of the publisher on which the message is sent. Accessing the Internet through multiple wired/wireless interfaces (e.g., a fiber optics modem connected through Ethernet plus a 5G modem connected over Wi-Fi) can improve dependability tangibly also for what concerns those portions of the paths located near the end points (publishers and subscribers). This is a complex, yet appealing aspect, which will be left as future work.

The tuple $\langle \text{SeqNo}, \text{PubID}, \text{PathID}, \text{IfcID} \rangle$ encodes all the information required by RMQTT. In particular, two copies refer to the same original message iff their SeqNo and PubID coincide. Hence, they enable deduplication on the subscribers' side. Instead, PathID and IfcID permit to uniquely identify the path on which every copy travelled, and are used for correctly gathering statistics on communication quality.

B. Adaptive Path Selection

As shown in [16], dependable MQTT communication can be achieved by statically selecting two or more distinct paths, which are operated according to RMQTT rules. As pointed out in that article, increasing the number of parallel paths on which a data stream is transported improves reliability. In fact, the likelihood that any given piece of data is delivered to destination always improves, since messages are lost only if they are dropped on all the paths of the redundant connection. Benefits are also obtained for latency, because what is delivered to applications is always the fastest copy of every message. This means that we can expect shorter end-to-end notification delays, which is particularly relevant for real-time data streams, including nonperiodic ones that convey asynchronous events (warning and error conditions).

However, there are practical limitations in leveraging the degree of redundancy to improve communication. In fact, every redundant path consumes network bandwidth and causes processing overheads on the publisher, subscribers, and intermediate brokers. The latter may become the bottleneck of the redundant communication infrastructure, and delays they cause on data streams may void in part the benefits of redundancy.

The solution we propose resembles the "n choose m" approach described in [16], where a communication infrastructure was considered that consist of a pool $\mathcal{B} = \{b_k, k \in [1, |\mathcal{B}|]\}$ of brokers geographically spread over a large area (e.g., a continent or the entire World) and connected to the Internet. Given the end points and assuming that every client accesses the Internet

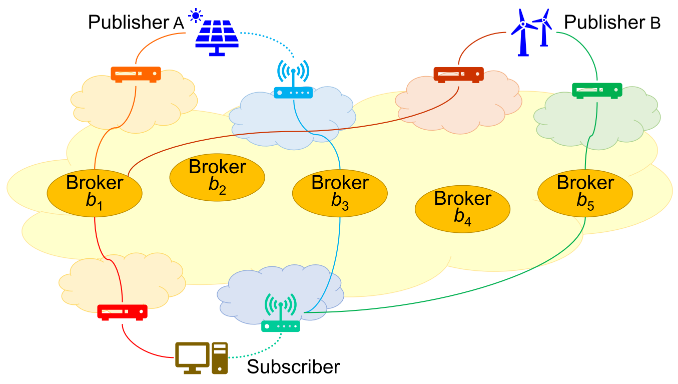


Fig. 1. Redundant MQTT architecture (two out of five brokers).

through a single interface, i.e., one Internet service provider (ISP), every *available path*, we denote π_k, is uniquely identified by the broker b_k it traverses. For this reason we use $\mathcal{B}$ to refer to the set of available paths as well. Then, m *active paths* were selected statically out of the $|\mathcal{B}|$ available ones, which are exploited by RMQTT. This subset is denoted $\mathcal{B}_\text{A} = \{\pi_{a_i}, i \in [1, m]\} \subseteq \mathcal{B}$, where sequence (a_i) describes a specific path selection pattern. Overall, there are $\binom{|\mathcal{B}|}{m}$ combinations of paths. For example, when $\mathcal{B} = \{b_1, b_2, b_3, b_4, b_5\}$ and $m = 2$, the number of possible choices equals 10, that is $\{b_1+b_2, b_1+b_3, b_1+b_4, b_1+b_5, b_2+b_3, b_2+b_4, b_2+b_5, b_3+b_4, b_3+b_5, b_4+b_5\}$, where the "$+$" symbol denotes path redundancy.

As we will see, larger values of m achieve better dependability, but they also imply higher bandwidth consumption. Generally speaking, a tradeoff has to be found about m by system designers, which depends on the quality of service (QoS) end-users ask for. In most practical contexts, increasing m beyond two (i.e., employing more-than-duplex redundancy) is likely to cause an excessive consumption of network resources, and should be avoided unless utmost dependability is demanded. For this reason, in the following we focused on the case $m = 2$. What we can do to get the most by RMQTT in spite of this limitation is to select the pair of active paths for every connection dynamically. An adaptive mechanism can be devised that, at any time:

1) evaluates the quality of communication of every path available in $\mathcal{B}$ by performing suitable measurements;
2) starting from the observed quality, selects two paths that will actively convey the redundant MQTT streams.

Many options exist for both of the above operations. Concerning communication quality evaluation, it is reasonable to define a metric that accounts for both message losses and delays on the different paths. Concerning path selection, although the most obvious choice is to consider the best m paths according to the selected metric, other choices are indeed possible. Such functions can be profitably seen as belonging to the control plane.

Fig. 1 describes a "2 out of 5" architecture. In the most general case, every path between a client and a broker is set through a distinct ISP. It is clearly possible to use for clients a single Internet access (as we did in the experiments below, since a fast

5G URLLC connection was not available), but doing so offers a diminished dependability.

C. Evaluating the Communication Quality of Paths

Because of the PubSub communication model MQTT relies on, information to evaluate quality metrics are mostly collected on the subscribers' side. The number of lost messages on every active path can be determined using counters, by checking the sequence numbers included in the messages themselves. Similarly, timestamps can be acquired upon message reception. To lower network usage, such quantities are collected locally and then delivered periodically to an entity denoted the *pool coordinator*, e.g., a centralized server that is in charge of managing the pool $\mathcal{B}$ of available brokers, taking care of aspects like load balancing, statistics on transmission latency, and diversity of paths. These activities belong to the control plane of our architecture: although relevant, they are part of a global, holistic optimization procedure, and will be specifically faced in future works.

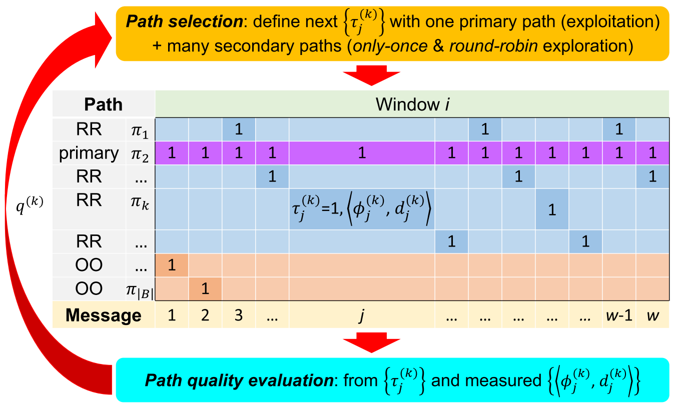
Publishers feed information to the control plane, too. Having publishers and subscribers time-synchronized, e.g., through NTP, can be useful to evaluate end-to-end latency, although not strictly necessary. In this article we consider the relative delays of paths as seen by every subscriber, which are trivially given by the offsets among the arrival times of the different copies of the same message. In fact, it can be safely assumed that departures from the publisher take place at about the same time (process data are typically small-sized).

D. How to Select Active Paths

The pool coordinator analyzes the information provided by clients and cyclically produces a feedback to let them know which paths to use. Always selecting the m paths with the highest quality (whatever this means, specific criteria must be defined) may not be the best option. In fact, quality evaluation is only possible for those paths that are being actively used. If paths that are deemed unworthy were no longer evaluated, the ability to dynamically track their condition would be disrupted.

The solution we envisage, described schematically in Fig. 2, resembles *reinforcement learning* (RL). The path with the best quality is always considered active, and is used for *exploitation*. We will call it the *primary* path π_{P}, even though the duplication–deduplication mechanism foreseen by seamless redundancy on the data plane is symmetric concerning path usage. When $m = 2$ the other active path is denoted *secondary* path π_{S}. It is employed for *exploration*, by iteratively selecting for every new message one of the nonbest paths. Parallel execution of exploitation and exploration on distinct paths maximizes dependability, yet retaining the capability of RL solutions to track the optimum configuration. Doing so doubles resource usage with respect to plain MQTT. However, as said before our solution is conceived for critical infrastructures, where service outages must be as close to zero as possible, otherwise there can be serious consequences.

Thanks to the above arrangement, applications layered above RMQTT are always provided a QoS that, at worst, matches what MQTT achieves on the previously evaluated best path. In the case communication on the primary path suddenly and unexpectedly deteriorates, possibly to the point that messages are dropped, worsening is mitigated by the contextual transmission of every message on the secondary path. The worst conditions occur when the primary path fails and a bad-behaving secondary path is being explored. To lower the probability of such events, heuristics can be additionally employed to reduce the amount of exploration carried out on those paths that, recently, proved to be unreliable.



Path selection: define next $\{\tau_j^{(k)}\}$ with one primary path (exploitation) + many secondary paths (*only-once* & *round-robin* exploration)
Path
Window *i*
RR π_1
primary π_2
RR ...
RR π_k
RR ...
OO ...
OO $\pi_{|B|}$
$q^{(k)}$
$\tau_j^{(k)}=1, \langle \phi_j^{(k)}, d_j^{(k)} \rangle$
1
Message
1 2 3 ... j ... ... w-1 w
Path quality evaluation: from $\{\tau_j^{(k)}\}$ and measured $\{\langle \phi_j^{(k)}, d_j^{(k)} \rangle\}$

Fig. 2. Adaptive seamless redundancy based on reinforcement learning (path quality evaluation and selection made on consecutive windows).

III. Experimental Evaluation

An experimental campaign was carried out on a real testbed to assess the performance of adaptive seamless redundancy (ASR) applied to MQTT. As we aim to show that communication is faster and more reliable than conventional MQTT, we mostly focused on the data plane and relied on some simplifying assumptions to make the control plane as simple as possible, yet retaining its basic functions. In particular, we considered PubSub communications between one publisher and one subscriber, which means that the path selection procedure must optimize communication quality of a single end-to-end connection.

The case of multiple subscribers/publishers using the same topic demands for some tradeoff. For example, either the average or worst-case quality, evaluated considering all the subscribers to the topic, should be taken into account as the metric for selecting the primary path. To provide more degrees of freedom, the limit on the number m of active brokers could be increased to three or more. To conserve bandwidth, exploitation is still performed on a single path, whereas exploration involves all the other paths, taken $m - 1$ at a time. Clearly, algorithms and heuristics in this case are noticeably more complex.

A. Testbed

Given this simple experimental setup, there is no need for a full-fledged control plane with a pool coordinator. Instead, the subscriber periodically sends its log about the received messages and their timestamps directly to the publisher, which evaluates communication quality and selects the primary and secondary paths. When a centralized entity collects and redistributes such information over the network, nonnegligible delays may affect such feedback, which reduce the reactivity of the adaptive selection mechanism and hence its effectiveness. This aspect will we analyzed in Section IV-C.

To carry out a realistic assessment, the testbed includes five brokers (Eclipse Mosquitto [18] version 1.6.9), executing in virtual machines hosted in the Amazon Web Services cloud and located in London (b_1), Frankfurt (b_2), Ireland (b_3), Stockholm (b_4), and Paris (b_5). Virtual machines are of type *t2.micro*, with one virtual CPU and 1 GB of RAM. Then, we developed two applications (one publisher and one subscriber), implemented using the `paho-mqtt` [19] library for the Python language (version 1.6.1), hosted on a PC running the Linux OS (Ubuntu 20.04 LTS) and deployed in the main Campus of the Politecnico di Torino, which communicate through RMQTT using above brokers. By exploiting sequence numbers it is possible to count how many messages went lost, while timestamps acquired on end nodes permit to evaluate the transmission latency for every message.

To simplify prototype implementation, publisher and subscriber were hosted on the same device, and exchanged information about path quality internally, without involving the network. By doing so computation of latency is eased, as timestamps are taken with the same clock. This is not a particularly limiting choice, as each copy of every data message travels on the related path from the publisher to the broker (located far away) and then back to the subscriber. The two halves of every such path (back and forth) are very likely the same (we do not know how routing tables are set at any time), just traversed in opposite directions. While leading to some in-path correlation, this does not affect tangibly the comparison among different paths.

B. Path Quality Evaluation

Evaluation of path quality is performed analyzing the messages exchanged in subsequent time windows, whose duration was set in such a way to include the generation of w messages. Let $\mathcal{T} = \{\tau_j^{(k)}, j \in [1, w], k \in [1, |\mathcal{B}|]\}$ be the path selection matrix, whose elements are Boolean values that specify if a copy of the jth message in the window has to be transmitted on π_k, and $n_\mathrm{T}^{(k)} = \sum_{j\in[1,w]} \tau_j^{(k)}$ the number of messages sent on that path. Since $\sum_{k\in[1,|\mathcal{B}|]} \tau_j^{(k)} = m = 2$, $\forall j$, the overall number of message *copies* sent on paths π_P and π_S in every window is $2w$, but not necessarily all of them reach the destination. In the worst (extremely unlikely) case, no messages at all are received.

On message arrival, the subscriber collects its sequence number and takes a timestamp. At the end of every window this information is sent to the publisher, which can determine the transmission outcome $\phi_j^{(k)}$ (1 for failure and 0 for success) of any given message on every path π_k on which it was actually sent. From it, the number of lost messages is determined as $n_\mathrm{L}^{(k)} = \sum_{j\in[1,w]|\tau_j^{(k)}=1} \phi_j^{(k)}$. For those messages that were received by the subscriber (i.e., $\phi_j^{(k)} = 0$) the measured latency $d_j^{(k)}$ can be computed from timestamps. Statistics for π_P ordinarily include many more samples than any path mapped on π_S, but this is not a severe issue.

For any path π_k, let $\overline{\phi}^{(k)} = n_\mathrm{L}^{(k)}/n_\mathrm{T}^{(k)}$ be the the fraction of lost messages and $\overline{d}^{(k)}$ the average latency, estimated from information about the past time window. A sensible choice for communication quality on π_k is to use a combined metric $q^{(k)}$ defined as

$$q^{(k)} = \overline{\phi}^{(k)} \cdot \tilde{D}_\mathrm{L} + \left(1 - \overline{\phi}^{(k)}\right) \cdot \overline{d}^{(k)}. \qquad (1)$$

The lower $q^{(k)}$, the higher the perceived path quality.

Parameter $\tilde{D}_\mathrm{L}$ in the leftmost term represents the *equivalent latency for lost messages*, and accounts for the fact that, from the applications' viewpoint, messages exceeding their deadline cause similar effects as those that went lost. We set this parameter to quite high values, well above $\max_{k\in[1,|\mathcal{B}|]}(\overline{d}^{(k)})$ and of the same order of magnitude as the worst-case latency we observed on paths in the experiments. In theory, latency might exceed $\tilde{D}_\mathrm{L}$ for some packet, but this is a very rare event. The rightmost term in (1) can be evaluated as $\frac{1}{n_\mathrm{T}^{(k)}} \sum_{j\in[1,w]|\tau_j^{(k)}=1\wedge\phi_j^{(k)}=0} d_j^{(k)}$, where the sum is restricted to the correctly delivered messages. This quantity can be computed also when all messages sent on π_k are lost and $\overline{d}^{(k)}$ is not defined.

C. Path Selection

The path selection procedure computes matrix $\mathcal{T}$, that is, the specific pair $\langle\pi_\mathrm{P}, \pi_\mathrm{S}\rangle$ for every message in the next window. It operates as follows: As soon as path metrics $\{q^{(k)}\}$ become available to the publisher at the end of the current window, the set of available paths is split in two disjoint subsets, *reliable* ($\mathcal{R}$) and *unreliable* ($\mathcal{U}$), depending on whether at least one message was received on π_k ($\mathcal{R} \cup \mathcal{U} = \mathcal{B}$, $\mathcal{R} \cap \mathcal{U} = \emptyset$).

If $|\mathcal{R}| \geq 2$, paths in $\mathcal{R}$ are then sorted in increasing q order. The first element is selected as the best path π_P and used for exploitation throughout the whole window, while exploration is performed on $\mathcal{B} \setminus \pi_\mathrm{P}$ in different ways for $\mathcal{R}$ and $\mathcal{U}$: paths in $\mathcal{R} \setminus \pi_\mathrm{P}$ (reliable paths with the exception of the best one) are considered in a round-robin fashion, whereas every path in $\mathcal{U}$ is explored only once. Such heuristic is meant to improve reliability further: in fact, exploration of paths on which no messages were received recently (that may consequently suffer from permanent failures) is limited to the bare minimum.

Conversely, exploitation takes place on the only reliable path when $|\mathcal{R}| = 1$, whereas it is performed on the same path selected in the previous windows when $|\mathcal{R}| = 0$ (i.e., all recent messages went lost on every path). In these two cases, exploration takes place according to a round-robin scheduling on the paths in $\mathcal{U} \setminus \pi_\mathrm{P}$ (that here corresponds to $\mathcal{B} \setminus \pi_\mathrm{P}$).

Above procedure is described by the pseudo code in Fig. 3, where sets $\mathcal{S}_\mathrm{OO}$ and $\mathcal{S}_\mathrm{RR}$ denote the paths which are explored in only-once and round-robin manner, respectively, while $\pi_\mathrm{S}[j]$ specifies the secondary path to be used for message j. Thus, $\tau_j^{(k)} = 1$ if $\pi_k = \pi_\mathrm{P} \vee \pi_k = \pi_\mathrm{S}[j]$, otherwise $\tau_j^{(k)} = 0$.

```
if $|\mathcal{R}| \geq 2$ then
    $\pi_P$:= $\arg\min_{\pi \in \mathcal{R}} (q^{(\pi)})$;
    $\mathcal{S}_{OO}$:= $\mathcal{U}$      // paths explored only once
    $\mathcal{S}_{RR}$:= $\mathcal{R} \setminus \pi_P$ // paths explored round-robin
else
    if $|\mathcal{R}| = 1$ then $\pi_P$:= $\mathcal{R}$[1];
    $\mathcal{S}_{OO}$:= $\emptyset$     // no paths explored only once
    $\mathcal{S}_{RR}$:= $\mathcal{U} \setminus \pi_P$ // all paths explored round-robin
j:=1; i:=1;
while i$\leq |\mathcal{S}_{OO}|$ do  // initial part of window
    $\pi_S$[j++]:= $\mathcal{S}_{OO}$[i++];
i:=1;
while j$\leq$w do       // remaining part of window
    $\pi_S$[j++]:= $\mathcal{S}_{RR}$[i++];
    if i> $|\mathcal{S}_{RR}|$ then i:=1;
```

Fig. 3. Pseudo-code for the path selection procedure.

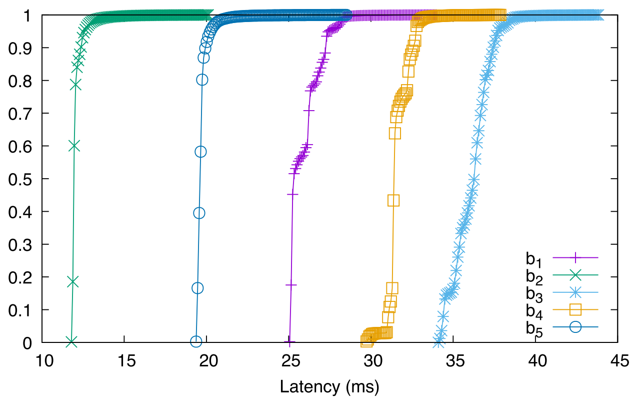


Fig. 4. CDF of latency for single paths b_1, b_2, b_3, b_4, and b_5.

IV. Results

In the experimental campaign the publisher cyclically generated one million asynchronous notifications at a rate of about 5 Hz (period $T = 199\,\text{ms}$), which were collected by the subscriber. Overall, the experiment lasted more than 55 h. By observing statistics about communication quality collected on the five paths in our testbed, two aspects were clear:

1) no messages went lost in the experiment: this is unsurprising, as losses in TCP are mainly due to events like link/router failures or severe DDoS attacks, whose occurrence is uncommon over limited time periods. Hence, we cannot realistically expect that they can be captured over a few days (many months, and even years would be needed);
2) the statistic distributions of latency on the different paths were quite narrow, as shown in Fig. 4, which reports their cumulative distribution function (CDF): this means that, most of the times, the fastest message copy is the one sent on the path with the lowest mean latency. However, it is worth remarking that in the captured log a fair number of exceptions were observed.

Logs acquired from the testbed are suitable to characterize the behavior of RMQTT and ASR in *normal* operating conditions. However, they are unable to describe what happens to the interconnected system when serious problems arise (natural and human-induced disasters, physical and cyber attacks, etc.). For this reason we artificially generated a second dataset from real data, where messages are sometimes lost. It can be used to characterize *critical* operating conditions.

Practically, a random pattern was generated for every path, which consisted in 1 000 000 Boolean values, and superposed to the related experimental log with the purpose to cull part of the messages, thus modeling losses. Since the events we want to describe are not occasional message losses, but outages that take a certain time to be recovered (think, e.g., to a broken router that once discovered triggers a change to the routing tables of other routers to restore connectivity), we modeled culling using a simplified Gilbert–Elliott model (and not a Bernoulli process). When in the *good* state no losses occur for MQTT messages on the given path, whereas in the *bad* state all of them are dropped.

Transition probabilities were set so that the mean sojourn time in the bad state is equal to $300\,T$ (about 60 s, a very pessimistic value for the reconfiguration time of routing tables), whereas for the good state it is nine times higher (9 min, path uptime before a new fault is experienced). As a consequence, about 10% of the messages went lost in theory on every path (9.61% in practice). This is a totally unrealistic behavior in ordinary conditions. However, we wished to model what may happen in a critical situation, e.g., when a number of deliberate attacks impact several portions of the Internet in such a way that every available path is repeatedly affected while the attack is underway. Assuming that failures on distinct paths are uncorrelated (this approximation is acceptable if paths are mostly disjoint), the probability that a message sent on an a duplex RMQTT connection is lost is, in theory, $\sim 1\%$.

A. Normal Operating Conditions

Diagrams in Fig. 5 depict some statistics about end-to-end notification latency, i.e., mean value and some percentiles, grouped by the considered testbed configuration. Bars in every group are shown this way only for clarity: actually, the bar related to any specific statistical quantity stretches down to the x axis. On the left side single paths are shown, followed by RMQTT connections exploiting two, three, four, and five statically selected paths, like in [16]. On the right side, results concern the adaptive path selection procedure with two concurrent active paths. Several cases were considered, by varying the width w of the window (24, 64, and 204 messages) and the penalty $\tilde{D}_L$ for lost messages (two values were used, 1 and 10 s, which have to be checked against the worst-case latency experienced in the tests, about 4 s).

As can be seen, mean latency and variability for single paths can be quite high. The fastest path is b_2, whose mean latency was $12.12\,\text{ms}$. Generally speaking, latency improves when the number of redundant paths increases. Although hardly noticeable from the figure, this also applies to configurations that include b_2. For example, mean latency decreased down to $12.07\,\text{ms}$ when five redundant paths are exploited (labeled *all* in the figure). This is because, although the fastest copy almost always came from b_2, it arrived on b_1, b_3, b_4, and b_5 (the second best path, on average) for 16, 29, 3, and 798 messages, respectively. Much

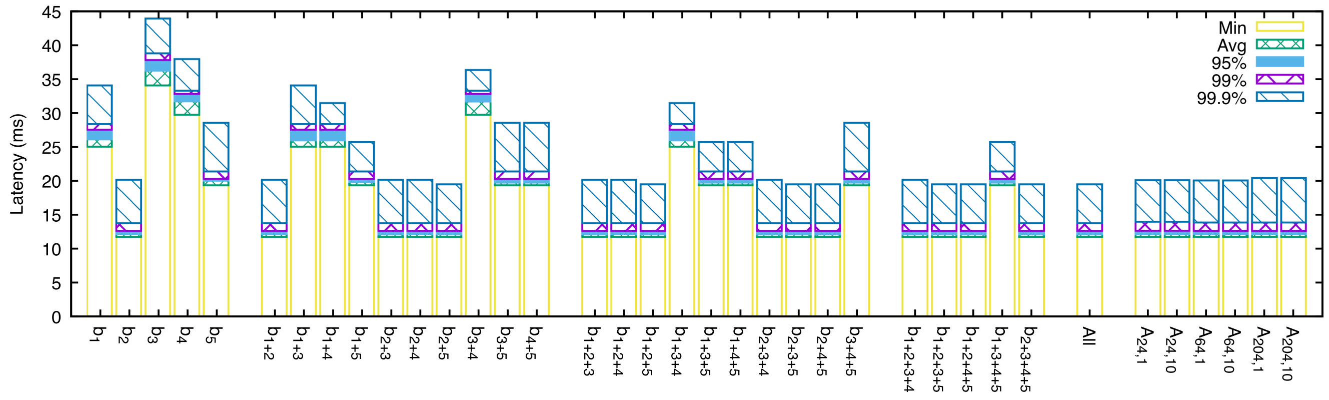


Fig. 5. Statistics on latency (normal conditions, no message losses): single paths, static multiple redundancy, and adaptive duplex redundancy.

sharper were the effects on the measured worst-case latency, which shrank from 4.012 s for b_4 (3.718 s for b_2) down to 2.091 s for $b_1+b_2+b_3+b_4+b_5$ (the same value obtained on b_3, which is the path that, concerning other statistics, behaved worst). The same happened to the 99.9 percentile, which decreased from 43.83 ms for b_3 (20.10 ms for b_2) down to 19.44 ms, and to the standard deviation, which went from 17.71 ms for b_5 (6.86 ms for b_2) down to 3.62 ms.

Another interesting quantity is the number of messages that experienced a latency higher than the generation period T. This happened 219, 128, 80, 98, and 335 times for single paths b_1, b_2, b_3, b_4, and b_5, respectively. T was exceeded on exactly two paths at the same time by 134 messages, on exactly three paths by a single message, on exactly four paths by a single message, and on all paths by 19 messages. The latter case is only seemingly counter-intuitive, and likely denotes situations where delays took place in the common part of the paths (in either the Campus or the ISP network), which made redundancy ineffective. This means, e.g., that when static redundancy is considered with $m = 2$, deadline T was missed, overall, $\binom{2}{2} \cdot 134 + \binom{3}{2} \cdot 1 + \binom{4}{2} \cdot 1 + \binom{5}{2} \cdot 19 = 333$ times, counting all ten possible combinations of two paths. More specifically, 20 times for $b_1 + b_2$ and $b_2 + b_4$, up to 112 times for $b_1 + b_5$.

What we learned from these results is that: 1) quite dissimilar performance can be obtained by blindly selecting different brokers, and 2) not always increasing the redundancy degree above three is justified in terms of the related improvements.

Groups of bars concerning ASR are labeled "$A_{w,\tilde{D}_\mathrm{L}}$." As can be seen, varying the windows width w only brought negligible differences to results. Likely, this depends on the fact that path quality varied slowly over time, and the ability to quickly track variations of network conditions was, in this case, not so relevant. The penalty value $\tilde{D}_\mathrm{L}$ was completely irrelevant, since in this experiment there were no losses. Statistics on latency were remarkably good, and similar to what can be achieved with three static paths: the mean value did not exceed 12.09 ms, the 99.9 percentile was 20.35 ms at worst, and standard deviation stayed below 5.85 ms. Similarly, latency exceeded T 22 times for both $A_{24,1}$ and $A_{204,10}$.

Concerning inner operation of the adaptive technique, when, e.g., $w = 24$, the number of whole windows (with the exception of the first one, used for initialization) was 41665. In 41570 cases b_2 was exploited as the primary path, and this happened 4, 2, 1, and 88 times for b_1, b_3, b_4, and b_5, respectively. Since there were no losses, no paths were classified as unreliable ($\mathcal{U} = \emptyset$), and hence secondary channels were always explored round-robin.

B. Critical Operating Conditions

In this case, the most interesting performance indicator is the fraction of received messages seen by applications, as depicted in Fig. 6. We expressed this quantity in dB as $-10 \log_{10}$ of the message delivery ratio (one minus the message loss ratio). This resembles sort of an "attenuation" of the data flow produced by the publisher due to message losses (0 dB means that nothing is dropped). Consistent with the approximations due to the limited duration of the experiment, losses (attenuation) are close to the theoretical values: about 9.6% (0.44 dB), 0.92% (0.040 dB), 0.089% (0.0039 dB), and 0.0085% (3.7×10^{-4} dB) with one, two, three, and four static paths, respectively. No messages went lost in the experiment when all five paths were used. Although the dataset was created artificially by necessity (information is dropped very seldom when using TCP), the advantages of seamless redundancy are clear as long as message losses on different paths are independent.

When the ASR mechanism is exploited, message losses (attenuation) lain in the range from 0.0207% (9×10^{-4} dB) for $w = 24$ to 0.2056% (0.00894 dB) for $w = 204$. As can be seen, decreasing the windows width w now achieves higher dependability, since the ability to promptly track variations in the network conditions (losses, in this particular case) improves. On the downside, more information is exchanged on the control plane. Penalty $\tilde{D}_\mathrm{L}$, which affects path selection by changing the computation of metrics $\{q_k\}$, had a limited impact: when $w = 24$ the same primary path was always chosen, irrespective of the value set for $\tilde{D}_\mathrm{L}$ (1 s or 10 s), whereas when $w = 204$ a discrepancy was observed in five windows out of 4900.

Concerning latency, one may (wrongly) expect that it does not vary with respect to the normal conditions, as the culling procedure applied to the experimental dataset, which randomly discards about 10% of the samples for each path, operates *independently* from actual transmissions on the testbed (which

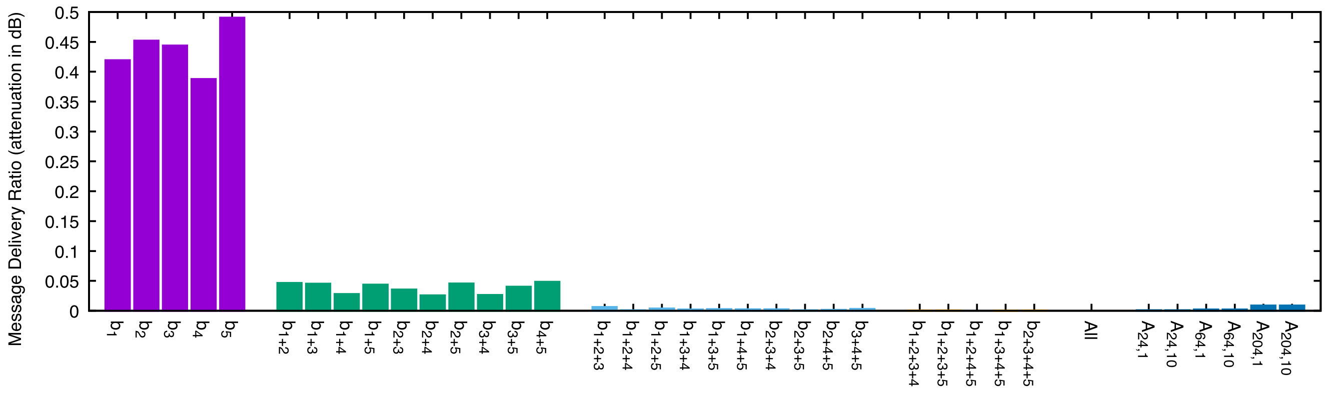


Fig. 6. Message delivery ratio (critical conditions, many message losses): single paths, static multiple redundancy, and adaptive duplex redundancy.

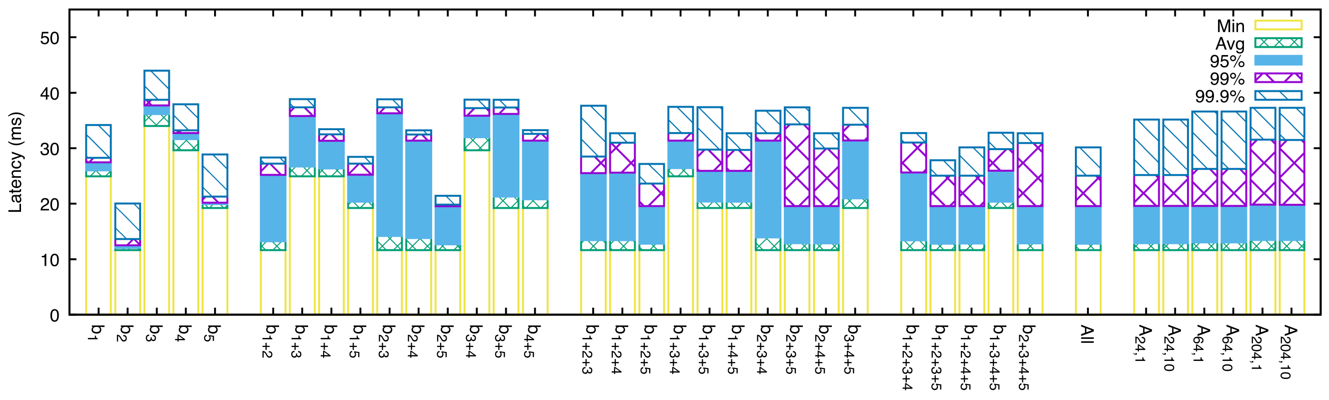


Fig. 7. Statistics on latency (critical conditions, many message losses): single paths, static multiple redundancy, and adaptive duplex redundancy.

determine latency). However, this is not the case: in fact, it may happen that messages that went lost on the fastest path are successfully delivered on a slower one, which enlarges latency. This is shown in the diagram in Fig. 7 where, e.g., the mean latency on all paths is 12.89 ms, whereas it ranges from 12.98 to 13.55 ms for ASR. These values are higher than 12.12 ms experienced for b_2, for which 9.88% of the messages were, however, lost. This is a known behavior of seamless redundancy, and does not depend on the adaptive path selection mechanism. It is not felt to be an issue, as the fraction of messages that exceed any given deadline on a redundant path is always lower than on every one of the paths that make it up (when considered alone). For instance, the number of received messages that experienced a delay larger than T on single paths is 211, 117, 75, 93, and 324 for b_1, b_2, b_3, b_4, and b_5, respectively, whereas it was 19 when all paths were included. When ASR was employed it ranged from 24 when $w = 24$ and $\tilde{D}_{\mathrm{L}} = 1$ s (in addition to 207 losses) to 27 when $w = 204$ and $\tilde{D}_{\mathrm{L}} = 1$ s (in addition to 2056 losses).

Analyzing inner ASR operation, when $w = 24$ b_2 was selected 37 440 times as the primary path, while b_1, b_3, b_4, and b_5 were exploited 420, 3, 43, and 3759 times, respectively. Instead, when $w = 204$ and $\tilde{D}_{\mathrm{L}} = 1$ s, path b_2 was selected as primary 4232 times, while b_1, b_3, b_4, and b_5 were exploited 95, 1, 15, and 557 times, respectively. Setting $\tilde{D}_{\mathrm{L}} = 10$ s changed the number of times path b_2, b_4, and b_5 were exploited to 4227, 16, and 561, respectively. Overall, results for ASR are much better than duplex static redundancy, and comparable to triplex redundancy. In other words, the same dependability can be attained by saving one third of the network bandwidth and computing resources on brokers (two versus three parallel paths).

C. Real Implementations

In our simplified testbed, where MQTT connection end-points are hosted on the same device, both quality evaluation and path selection take place contextually at the beginning of every time window. This is not a prerequisite for the correct operation of the adaptive mechanism. Since path evaluation in real (distributed) implementations unavoidably takes some time, selection is performed on slightly outdated quality metrics. Latency with which the control plane provides feedback to the data plane does not impact on the correct RMQTT behavior. However, it may worsen its performance, by negatively affecting the QoS end nodes experience. Generally speaking, the closer (in time) evaluation and selection, the better the ability to track variations of network conditions, which implies that prompt operations are sought for the control plane.

To analyze the effect of such delays on dependability, an additional postanalysis was performed for ASR by assuming a slower control plane. In particular, information about broker

TABLE I
STATISTICS ON LATENCY (AVG/STD, PERCENTILES, MIN/MAX) AND SUCCESS RATIO – NORMAL/CRITICAL CONDITIONS – STATIC RMQTT/ASR

	Normal conditions (no message losses)								Critical conditions (many message losses)								
Approach	Min	Avg	p_{90}	p_{95}	p_{99}	$p_{99.9}$	Max	Std	Succ. (%)	Min	Avg	p_{90}	p_{95}	p_{99}	$p_{99.9}$	Max	Std
b_1	24.97	26.04	27.25	27.45	28.30	33.98	3990.89	12.90	90.7963	24.97	26.06	27.25	27.49	28.30	34.18	3990.89	13.50
b_2	**11.71**	**12.12**	12.38	12.59	13.72	**20.10**	3718.38	6.86	90.1157	11.71	12.12	12.38	12.59	13.72	20.10	3718.38	7.17
b_3	34.00	***36.15***	37.44	37.72	38.73	***43.83***	**2091.04**	**4.56**	90.2823	34.00	36.13	37.42	37.71	38.73	43.96	2091.04	4.73
b_4	29.65	31.67	32.59	32.74	33.22	37.86	***4011.68***	6.84	91.4550	29.66	31.67	32.59	32.74	33.22	37.92	4011.68	7.11
b_5	19.28	19.96	19.94	20.24	21.32	28.50	3730.20	***17.71***	89.3181	19.28	19.98	19.94	20.24	21.33	28.91	3730.20	18.71
b_{1+2}	11.71	12.08	12.38	12.59	13.72	20.10	3718.38	6.04	98.9356	11.71	13.32	13.54	25.24	27.24	28.36	3718.38	7.29
b_{1+3}	24.97	25.90	27.25	27.45	28.30	33.98	2091.04	3.87	98.9641	24.97	26.75	28.23	35.82	37.38	38.83	2091.04	4.86
b_{1+4}	24.97	25.91	27.25	27.45	28.30	31.39	3990.89	6.46	99.3592	24.97	26.41	28.26	31.37	32.50	33.42	3990.89	6.84
b_{1+5}	19.28	19.76	19.94	20.24	21.32	25.65	3730.20	7.88	98.9942	19.28	20.40	23.83	25.27	27.25	28.47	3730.20	9.71
b_{2+3}	11.71	12.07	12.38	12.59	13.72	20.10	2091.04	3.68	99.1832	11.71	14.28	13.78	36.28	37.38	38.79	2091.04	7.99
b_{2+4}	11.71	12.08	12.38	12.59	13.72	20.10	3718.38	6.05	99.4093	11.71	13.91	14.13	31.38	32.48	33.22	3718.38	8.43
b_{2+5}	11.71	12.07	12.38	12.59	13.72	19.44	3718.38	6.05	98.9560	11.71	12.77	13.55	19.60	19.91	21.48	3718.38	6.67
b_{3+4}	29.65	31.63	32.59	32.74	33.22	36.25	2091.04	3.65	99.3919	29.66	32.00	32.87	35.88	37.24	38.76	2091.04	4.10
b_{3+5}	19.28	19.71	19.94	20.24	21.32	28.50	2091.04	3.65	99.0737	19.28	21.33	25.40	36.19	37.35	38.73	2091.04	6.22
b_{4+5}	19.28	19.72	19.94	20.24	21.32	28.50	3730.20	6.05	98.8923	19.28	20.87	22.84	31.38	32.61	33.27	3730.20	7.09
b_{1+2+3}	11.71	12.07	12.38	12.59	13.72	20.10	2091.04	3.63	99.8528	11.71	13.52	15.78	25.53	28.53	37.64	2091.04	5.80
b_{1+2+4}	11.71	12.08	12.38	12.59	13.72	20.10	3718.38	6.04	99.9759	11.71	13.51	17.61	25.66	31.02	32.72	3718.38	7.46
b_{1+2+5}	11.71	12.07	12.38	12.59	13.72	19.44	3718.38	6.02	99.9114	11.71	12.88	16.36	19.64	23.69	27.21	3718.38	6.54
b_{1+3+4}	24.97	25.90	27.25	27.45	28.30	31.39	2091.04	3.70	99.9515	24.97	26.45	28.32	31.39	32.75	37.44	2091.04	4.23
b_{1+3+5}	19.28	19.70	19.94	20.24	21.32	25.65	2091.04	3.61	99.9354	19.28	20.46	25.12	25.94	29.77	37.41	2091.04	4.38
b_{1+4+5}	19.28	19.71	19.94	20.24	21.32	25.65	3730.20	6.03	99.9421	19.28	20.42	25.12	25.95	29.73	32.72	3730.20	6.42
b_{2+3+4}	11.71	12.07	12.38	12.59	13.72	20.10	2091.04	3.64	99.9408	11.71	14.01	16.78	31.39	32.73	36.77	2091.04	6.93
b_{2+3+5}	11.71	12.07	12.38	12.59	13.72	19.44	2091.04	3.63	99.9760	11.71	12.99	17.61	19.64	34.31	37.37	2091.04	4.94
b_{2+4+5}	11.71	12.07	12.38	12.59	13.72	19.44	3718.38	6.02	99.9655	11.71	12.95	17.30	19.64	29.97	32.70	3718.38	6.75
b_{3+4+5}	19.28	19.71	19.94	20.24	21.32	28.50	2091.04	3.62	99.9249	19.28	21.02	31.04	31.41	34.26	37.26	2091.04	5.28
$b_{1+2+3+4}$	11.71	12.07	12.38	12.59	13.72	20.10	2091.04	3.63	99.9961	11.71	13.50	18.27	25.69	31.04	32.74	2091.04	5.68
$b_{1+2+3+5}$	11.71	12.07	12.38	12.59	13.72	19.44	2091.04	3.62	99.9847	11.71	12.89	17.86	19.64	25.11	27.87	2091.04	4.46
$b_{1+2+4+5}$	11.71	12.07	12.38	12.59	13.72	19.44	3718.38	6.02	100.0000	11.71	12.90	18.43	19.64	25.12	30.18	3718.38	6.55
$b_{1+3+4+5}$	19.28	19.70	19.94	20.24	21.32	25.65	2091.04	3.61	99.9971	19.28	20.42	25.12	26.00	29.86	32.82	2091.04	4.21
$b_{2+3+4+5}$	11.71	12.07	12.38	12.59	13.72	19.44	2091.04	3.62	99.9948	11.71	12.95	18.22	19.64	30.95	32.73	2091.04	4.64
All	**11.71**	**12.07**	12.38	12.59	13.72	**19.44**	**2091.04**	**3.62**	100.0000	11.71	12.89	18.43	19.64	25.12	30.18	2091.04	4.45
$A_{24,1}$	11.71	***12.09***	12.38	12.61	13.92	20.03	3718.38	5.83	99.9793	11.71	12.98	19.43	19.66	25.21	35.15	3718.38	6.49
$A_{24,10}$	11.71	12.09	12.38	12.61	13.92	20.03	3718.38	5.83	99.9793	11.71	12.98	19.43	19.66	25.21	35.15	3718.38	6.49
$A_{64,1}$	11.71	**12.08**	12.38	12.60	13.81	**20.00**	3718.38	5.82	99.9524	11.71	13.10	19.46	19.68	26.30	36.63	3718.38	6.64
$A_{64,10}$	11.71	12.08	12.38	12.60	13.81	20.00	3718.38	5.82	99.9524	11.71	13.10	19.46	19.68	26.30	36.63	3718.38	6.64
$A_{204,1}$	11.71	12.09	12.38	12.60	13.82	***20.35***	3718.38	5.85	99.7944	11.71	13.55	19.60	19.90	31.56	37.27	3718.38	7.13
$A_{204,10}$	11.71	12.09	12.38	12.60	13.82	20.35	3718.38	5.85	99.7992	11.71	13.55	19.60	19.89	31.49	37.26	3718.38	7.12
$A^{\mathrm{D}}_{24,1}$	11.71	12.09	12.38	12.61	13.93	20.15	3718.38	5.85	99.9504	11.71	13.10	19.46	19.68	26.67	36.67	3718.38	6.67
$A^{\mathrm{D}}_{24,10}$	11.71	12.09	12.38	12.61	13.93	20.15	3718.38	5.85	99.9504	11.71	13.10	19.46	19.68	26.67	36.67	3718.38	6.67
$A^{\mathrm{D}}_{64,1}$	11.71	12.08	12.38	12.60	13.81	20.00	3718.38	5.85	99.8661	11.71	13.38	19.52	19.76	31.48	37.30	3718.38	7.03
$A^{\mathrm{D}}_{64,10}$	11.71	12.08	12.38	12.60	13.81	20.00	3718.38	5.85	99.8661	11.71	13.38	19.52	19.76	31.48	37.30	3718.38	7.03
$A^{\mathrm{D}}_{204,1}$	11.71	12.09	12.38	12.60	13.82	20.38	3718.38	5.85	99.4745	11.71	14.05	19.68	25.21	35.71	37.53	3730.20	7.66
$A^{\mathrm{D}}_{204,10}$	11.71	12.09	12.38	12.60	13.82	20.38	3718.38	5.85	99.4795	11.71	14.05	19.68	25.21	35.55	37.53	3730.20	7.66

The bold values refer to relevant values (maximum, minimum) that are referred to in the text.

selection now becomes effective after one entire time window has elapsed (and not immediately, as in the previous case). Likely, this is a quite pessimistic hypothesis, but we are interested in assessing how much performance may worsen because of a delayed feedback. Results are reported at the bottom of Table I, where ASR with delayed feedback is labeled "$A^{\mathrm{D}}_{w,\tilde{D}_{\mathrm{L}}}$". Statistics for all redundancy approaches (static and adaptive) have been also reported for completeness.

As can be seen, negligible variations were obtained for latency in normal conditions (no losses, on the left side). Concerning critical conditions (many losses, on the right side), latency increase is more pronounced. Moreover, a sensible growth was experienced for message losses when the window size (and hence, the delay with which feedback is returned) is larger ($w = 204$), which, however, remained always below the duplex static case.

V. CONCLUSION

The ability to correctly deliver messages on unreliable networks with a reasonable certainty that they arrive within the given deadlines can be improved by resorting to seamless redundancy. When MQTT data streams are considered between end points located in distant places and interconnected through the Internet, an effective solution is to deploy a number of brokers and to rely on a redundant version of MQTT that performs duplication and deduplication of messages.

In this article, two approaches were considered that are based on RMQTT, and their performance compared through an experimental campaign on real devices communicating over long-range geographic connections. In the former, a number of active paths were selected statically before operations are started. Instead, the latter exploited an adaptive mechanism that relies on separate data and control planes. RMQTT is used at the data plane, whereas the control plane continuously analyses the communication quality of the available paths and dynamically selects which ones to use for transmission. The technique we adopted to this aim resembles reinforcement learning, where exploitation is performed on the path that is known to currently offer the best behavior while exploration is carried out on all the other available paths.

Results show two relevant achievements: first, seamless redundancy always provides a tangible boost to communication quality, by ensuring unprecedented levels of dependability for communication over the Internet, which fit the needs of many critical infrastructures demanding high resilience. Second, the adaptive path selection mechanism is able to offer high communication quality by using only a fraction of the bandwidth and network resources of the static redundancy approach. Hence, it provides end users with an inexpensive and mostly backward compatible way to support resilient remote monitoring and control applications in quasi-real time.

Future works will include more effective path selection mechanisms, also considering more-than-duplex adaptive redundancy, multiple subscribers, and taking path diversity into account.

References

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan.–Mar. 2004.

[2] J. Laprie, “From dependability to resilience,” in *Proc. 38th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2008, pp. G8–G9.

[3] “OPC UA specifications and information models,” Accessed: Sep. 30, 2022. [Online]. Available: https://reference.opcfoundation.org/

[4] Y. Li, J. Jiang, C. Lee, and S. H. Hong, “Practical implementation of an OPC UA TSN communication architecture for a manufacturing system,” *IEEE Access*, vol. 8, pp. 200100–200111, 2020.

[5] P. Denzler, T. Frühwirth, D. Scheuchenstuhl, M. Schoeberl, and W. Kastner, “Timing analysis of TSN-enabled OPC UA PubSub,” in *Proc. IEEE 18th Int. Conf. Factory Commun. Syst.*, 2022, pp. 1–8.

[6] C. Zunino, G. Cena, S. Scanzio, and A. Valenzano, “Experimental characterization of asynchronous notification latency for subscriptions in OPC UA,” in *Proc. IEEE 26th Int. Conf. Emerg. Technol. Factory Automat.*, 2021, pp. 01–08.

[7] R. Gupta, F. Sossan, and M. Paolone, “Grid-aware distributed model predictive control of heterogeneous resources in a distribution network: Theory and experimental validation,” *IEEE Trans. Energy Convers.*, vol. 36, no. 2, pp. 1392–1402, Jun. 2021.

[8] “MQTT specification,” Accessed: Sep. 30, 2022. [Online]. Available: https://mqtt.org/mqtt-specification/

[9] J. M. Ramadhan, R. Mardiati, and I. N. Haq, “IoT monitoring system for solar power plant based on MQTT publisher / subscriber protocol,” in *Proc. 7th Int. Conf. Wireless Telematics*, 2021, pp. 1–6.

[10] OASIS, “Advanced Message queuing protocol (AMQP) version 1.0,” 2012. Accessed: May 4, 2023. [Online]. Available: http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf

[11] M. Topp, “OPC UA + MQTT = A popular combination for IoT expansion,” Accessed: Sep. 30, 2022. [Online]. Available: https://opcconnect.opcfoundation.org/2019/09/opc-ua-mqtt-a-popular-combination-for-iot-expansion/

[12] *IEC 62439-3:2021, Industrial Communication Networks - High Availability Automation Networks - Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR)*, IEC Standard 62439-3:2021, 2021.

[13] M. Rentschler and P. Laukemann, “Towards a reliable parallel redundant WLAN black channel,” in *Proc. IEEE 9th Int. Workshop Factory Commun. Syst.*, 2012, pp. 255–264.

[14] J. M. Rhee, I. R. A. Altaha, D. N. M. Hoang, D. H. Kim, J. S. Yang, and S. Y. Park, “Seamless CAN: CAN bus plus high-availability seamless redundancy,” in *Proc. Int. Conf. Inf. Commun. Technol. Convergence*, 2021, pp. 1284–1286.

[15] M. Popovic, M. Mohiuddin, D.-C. Tomozei, and J.-Y. L. Boudec, “iPRP–the parallel redundancy protocol for IP networks: Protocol design and operation,” *IEEE Trans. Ind. Inform.*, vol. 12, no. 5, pp. 1842–1854, Oct. 2016.

[16] C. Zunino, D. Malena, G. Cena, S. Scanzio, and A. Valenzano, “Resilient MQTT to support uninterruptible services,” in *Proc. IEEE 18th Int. Conf. Factory Commun. Syst.*, 2022, pp. 1–8.

[17] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, “TCP extensions for multipath operation with multiple addresses,” Internet Eng. Task Force, Wilmington, DE, USA, RFC 8684, Mar. 2020.

[18] “Eclipse mosquitto,” Accessed: Sep. 30, 2022. [Online]. Available: https://mosquitto.org

[19] “Paho MQTT library,” Accessed: Sep. 30, 2022. [Online]. Available: https://www.eclipse.org/paho/

Claudio Zunino (Member, IEEE) received the degree in computer engineering and the Ph.D. degree in software engineering from Politecnico di Torino, Turin, Italy, in 2000 and 2005, respectively.

Since 2023, he has been a Senior Researcher with the Institute of Electronics and Computer and Telecommunications, National Research Council of Italy, Turin, Italy. He has authored or coauthored numerous papers in international journals and conferences in the areas of his interest. His research interests include wireless communication, industrial Ethernet protocols, computer graphics, parallel and distributed computing, and scientific visualization.

Dr. Zunino serves as a Reviewer for various international conferences and journals, and has participated in the program and organizing committees of many international conferences focused on industrial informatics and automation. He was also a Co-Guest Editor for the Special Section on Industrial Communication Technologies and Systems published in the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS and he is currently an Associate Editor for the IEEE ACCESS journal.

Gianluca Cena (Senior Member, IEEE) received the M.S. degree in electronic engineering and the Ph.D. degree in information and system engineering from the Politecnico di Torino, Turin, Italy, in 1991 and 1996, respectively.

Since 2005, he has been a Director of Research with the Institute of Electronics, Information Engineering and Telecommunications, National Research Council of Italy, Turin, Italy. He has coauthored about 160 papers and one patent in the areas of his interests. His research interests include wired and wireless industrial communication systems, real-time protocols, and automotive networks.

Dr. Cena was the recipient of the Best Paper Award of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, in 2017, and the IEEE Workshop on Factory Communication Systems, in 2004, 2010, 2017, 2019, and 2020. Since 2009, he has been serving as an Associate Editor for the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. He served as Program Co-Chairman of the IEEE Workshop on Factory Communication Systems, in 2006 and 2008, and as Track Co-Chairman in six editions of the IEEE Conference on Emerging Technologies and Factory Automation.

Stefano Scanzio (Senior Member, IEEE) received the laurea and Ph.D. degrees in computer science from Politecnico di Torino, Turin, Italy, in 2004 and 2008, respectively.

Since 2009, he has been with the National Research Council of Italy, Rome, Italy, where he is currently a Senior Researcher with the institute CNR-IEIIT, Turin, Italy. He teaches several courses on computer science wth Politecnico di Torino. He has coauthored more than 90 papers in international journals and conferences in the areas of his interests. He took part in the program and organizing committees of many international conferences of primary importance in his research areas. His research interests include industrial communication systems, real-time networks, wireless networks, and artificial intelligence.

Dr. Scanzio was the recipient of the 2017 Best Paper Award of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS and four Best Paper Awards in 2010, 2017, 2019, and 2020 editions of IEEE WFCS. He is currently an Associate Editor for the *Ad Hoc Networks* (Elsevier), IEEE ACCESS journals, and *Electronics* (MDPI).

Adriano Valenzano (Senior Member, IEEE) received the laurea degree magna cum laude in electronic engineering from Politecnico di Torino, Torino, Italy, in 1980.

He has been a Director of Research with the National Research Council of Italy (CNR), Rome, Italy, since 1991. He is currently a Research Associate with the Institute of Electronics, Computer, and Telecommunication Engineering, Torino, Italy, where he carries out studies on distributed computer systems, local area networks, and communication protocols. He has coauthored approximately 200 refereed journal and conference papers in the area of computer engineering.

Dr. Valenzano was the recipient of the 2013 IEEE IES and ABB Lifetime Contribution to Factory Automation Award. He was also the recipient of the best paper published in the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, during 2016, and the Best Paper Awards for the papers presented at the 5th, 8th, 13th, 15th and 16th IEEE Workshops on Factory Communication Systems (WFCS 2004, WFCS 2010, WFCS 2017, WFCS 2019, WFCS 2020). Since 2007, he has been serving as an Associate Editor for the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. He has served as a Technical Referee for several international journals and conferences, also taking part in the program committees of international events of primary importance.