

Application Protocols enabling Internet of Remote Things via Random Access Satellite Channels

Manlio Bacco, Marco Colucci, Alberto Gotta
Institute of Information Science and Technologies (ISTI)
National Research Council (CNR), via G. Moruzzi, 1, Pisa (Italy)
e-mails: {name.surname}@isti.cnr.it

Abstract—Nowadays, Machine-to-Machine (M2M) and Internet of Things (IoT) traffic rate is increasing at a fast pace. The use of satellites is expected to play a large role in delivering such a traffic. In this work, we investigate the use of two of the most common M2M/IoT protocols stacks on a satellite Random Access (RA) channel, based on DVB-RCS2 standard. The metric under consideration is the completion time, in order to identify the protocol stack that can provide the best performance level.

I. INTRODUCTION

Satellite-based M2M communications represent a large fraction of the IoT market, showing an increasing popularity both in the research and in the industrial community. The ubiquitous coverage provided by the satellites may represent a key feature to enable the so-called IoT massive internetworking, bringing the connectivity even in remote areas that are unlikely to be covered by other communication infrastructures (e.g., cellular).

In order to evaluate the feasibility of a satellite-based solution, along with any physical layer issues (e.g., Signal-to-Noise Ratio (SNR), power consumption), the interactions between the transport and the application layer protocols need further investigations. Connection-oriented transport protocols, like Transmission Control Protocol (TCP), require connection establishment procedures, the use of flow control or congestion control algorithms, which may increase the communication overhead. The latter is an issue that needs to be carefully taken into account, especially in the case of IoT/M2M short-lived connections. In order to mitigate the aforementioned issue, Internet Engineering Task Force (IETF) has proposed the use of Constrained Application Protocol (CoAP) (RFC 7252), a lightweight protocol designed for resource-constrained devices. It relies on the use of User Datagram Protocol (UDP) at the transport layer; because of this, the reliability is left out as an optional feature, to be implemented at the application layer. CoAP endpoints exchange messages according to a request/response mode and the resources are accessed through a Uniform Resource Identifier (URI). In order to avoid a polling mechanism, IETF has designed a protocol extension to CoAP, based on the so-called *observer* design pattern (RFC 7641). CoAP clients *register* to the CoAP server; then, each client receives a *notification* every time the state of a resource changes. The observer pattern is somewhat similar to the Publish / Subscribe (PUB/SUB) paradigm [1], as implemented by MQTT, for instance. The performance provided by the

use of MQTT on RA satellite channels, according to the Digital Video Broadcasting - Return Channel via Satellite, II generation (DVB-RCS2) standard [2], has been preliminary studied in [3]. Contrarily to CoAP, MQTT is TCP-based, thus the congestion control algorithm at the transport layer of each Return Channel Satellite Terminal (RCST) is responsible for rate control and retransmissions, if any erasures occur on the satellite channel.

In this work, we propose a comparison between the MQTT-based scenario in [3], and a CoAP-based protocol stack. The performance metric under consideration is the *completion time*, which is the time a producer takes to successfully deliver data to a consumer. The rest of this paper is organized as follows: Section II reviews some of the most relevant works in the literature, focusing on IoT/M2M communication scenarios via satellite and on the comparisons between IoT/M2M protocol stacks. Section III compares some typical M2M/IoT protocol stacks. Section IV describes the application scenario under consideration and Section V shows some preliminary numerical results obtained via extensive simulation runs. Finally, the conclusions are provided in Section VI.

II. RELATED WORKS

The work in [4] considers M2M terminals that communicate with a remote receiver via a satellite link. Each terminal transmits a fixed amount of data: RA is used to deliver the first few messages, then the Network Control Centre (NCC) allocates reserved timeslots to each RCST, in order to ensure a successful delivery of data. The authors assess the system performance by setting the burst duration and by using three different reception modes at the receiver: Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), Power Division Multiple Access (PDMA) and Turbo Code Division Multiple Access (TCDMA). The throughput and the packet error rate are the performance metrics under consideration. The authors show that TCDMA with multiuser detection outperforms both FDMA and TDMA in terms of throughput, thus minimizing the time required to serve a given set of M2M terminals. In [5], a satellite-based Wireless Sensor Network (WSN) is considered. In order to handle a potentially large number of M2M devices, the authors suggest to organize the nodes in clusters of different sizes. Within a cluster, the nodes communicate with the cluster-head (CH), which in turn forwards the collected data to the satellite gateway.

arXiv:1706.09787v1 [cs.NI] 29 Jun 2017

The clustering mechanism aims at organizing the clusters in such a way that the application requirements (e.g., minimum required SNR) can be met. Moreover, in order to cope with the rain fading that can affect the signal propagation, multiple interconnected satellite gateways are considered. Physical layer metrics are used to assess the performance level: SNR and energy consumption. The aforementioned works do not consider any interactions with higher layer protocols, focusing on Media Access Control (MAC) and physical layer metrics. Nonetheless, they consider M2M scenarios involving satellite communications.

Conversely, application-layer protocols are explicitly compared in [6], [7], [8]. In [6], three classes of protocols for M2M communications are compared: protocols targeting the Service-Oriented Architecture (SOA); protocols implementing the Representational State Transfer (REST) paradigm; and message-oriented protocols. As SOA protocol, the authors consider OPC¹ Unified Automation (UA), a platform-independent middleware, whereas CoAP and MQTT are chosen as representative of REST architectures and message-oriented protocols, respectively. The application scenario is represented by a cellular network delivering M2M data, where reliability and real-time data exchanges are required. The completion time in a emulated cellular network is used as key performance indicator, and, according to the authors, OPC-based communications outperform CoAP and MQTT-based data transfer, at the price of a larger overhead. MQTT is TCP-based, and the contributions provided in [9], [10], [11], [12] shed some lights on how TCP behaves in presence of a random access satellite link dominated by collisions, because it may represent a limiting factor on the achievable throughput in satellite environments. On the other side, CoAP is UDP-based, and a comparison is on order when dealing with random access satellite links, in order to provide some reference figures on the achievable performance level. A preliminary comparison between MQTT and CoAP is provided in [7], in terms of bandwidth usage and latency. In order to compare the two protocols, the authors consider a simple scenario composed by a single MQTT publisher that sends data to a MQTT broker; similarly, the CoAP-based scenario considers data exchanges between a Hypertext Transfer Protocol (HTTP) client and a CoAP server. The metrics under consideration are the total amount of generated traffic and the average Round-Trip Time (RTT). Both reliable and non-reliable data transmissions are considered. In both cases, when no losses occur, CoAP transfers less data and exhibits a shorter RTT than MQTT, on average. In [8], a satellite-based architecture is considered: multiple M2M devices send data to a remote gateway. The performance provided by the use of MQTT and of CoAP is evaluated, and the authors show that CoAP can be properly tuned, in order to outperform MQTT even in the presence of high offered traffic.

¹A description of the OPC standard is available at <https://opcfoundation.org/about/what-is-opc/>

III. TYPICAL IOT PROTOCOL STACKS

In [13], the authors propose a general satellite network architecture for IoT/M2M application scenarios, where the use of satellite communications could provide some benefits, such as broadcast communications, large coverage also in suburban and rural areas, support for highly mobile nodes in absence of the fixed infrastructure. When comparing possible IoT architectures, several different protocol stacks can be used, each providing different advantages. Two communication paradigms are typically considered in IoT scenarios: request/response and PUB/SUB ones. IoT nodes collect or produce new data in an event-driven or a time-driven fashion, typically. While the latter may exhibit regularity over time, the former shows variable traffic patterns. If the request/response paradigm is taken into account, the clients should periodically query the servers in order to retrieve fresh data. On high-delay links, like in the case of the satellites, the time needed to successfully complete data exchanges should be carefully evaluated. The PUB/SUB paradigm, a possible alternative to the request/response one, allows the data consumers, or *subscribers*, to receive any fresh data as soon as they are available at the data producers, or *publishers*. A key feature of the PUB/SUB paradigm is the decoupling between data producers and data consumers at an intermediate entity, called *broker*. In topic-based PUB/SUB systems, each data piece belongs to one or more *topics*, or logical channels. The publishers send new data to the broker, specifying the topic(s) the data belong to. The broker keeps the list of the active topics and, for each topic, the list of the active subscribers. Each subscriber, in fact, declares its interests to the broker through an initial *registration* procedure. After that, the mechanism is straightforward: the publishers send new data to the broker, which forward them to the subscribed nodes. On high-delay links, relying on a PUB/SUB-based data exchange allows halving the delivery delay than relying on a request/response-based one.

CoAP and MQTT are notable examples of application protocols implementing the aforementioned two paradigms: request/response the former, PUB/SUB the latter. MQTT and CoAP protocol stacks can be seen in Figure 1 and are discussed in Section III-A and III-B, respectively.

A. MQTT protocol

MQTT is a M2M/IoT application protocol designed by IBM in 1999 for use in satellite networks. Since then, its use has largely widespread to terrestrial communications. A typical MQTT data packet is composed of a 2 bytes long fixed header part, a variable header part whose size depends on the packet type, and a variable length payload. Each data packet is sent to the broker, which maintains the list of the active subscriptions and of the active topics. Although reliable data transmission are inherently guaranteed by TCP, MQTT offers three Quality of Service (QoS) levels to deliver the messages. In fact, TCP guarantees the reliability for the messages exchanged over the network connection between broker-publisher and broker-subscriber, but an End-to-End (E2E) mechanism is absent. To

MQTT	CoAP
TCP	UDP
IPv4, IPv6, 6LoWPAN	
MAC layer	
Physical layer	

Fig. 1: Typical IoT protocol stacks

address that, MQTT provides additional reliability levels at the application.

B. CoAP protocol

CoAP follows a REST architectural style and is designed for resource-constrained environments. Each CoAP server logically encapsulates a *resource*, uniquely identify by a URI. A CoAP client sends a request by means of a *Confirmable* or *Non-confirmable* message, in order to retrieve the resource representation available at the server. If a *Confirmable* message type is sent, an Acknowledgment (ACK) is expected to confirm the correct reception of data at the intended receiver; otherwise, an unreliable data exchange occurs (*Non-confirmable* message type). A CoAP request contains the URI of the resource and is typically performed by means of the HTTP *GET* verb. The typical message format includes a fixed-size header (4 bytes), a variable-length *Token* field (0-8 bytes), an *options* field, and the payload. CoAP is UDP-based, and it provides optional reliability at the application layer. A *NSTART*-long [packets] transmission window² is dictated by the specifications; in the default configuration, *NSTART* is equal to one. If *Confirmable* messages are sent, then the Automatic Repeat reQuest (ARQ) mechanism in use is a simple Stop-and-Wait protocol, employing exponential back-off. This choice is motivated by the fact that the protocol is intended for low-power resource-constrained devices, where the implementation of more complex mechanisms can present some computational or technological issues because of the limited available resources. Anyway, CoAP is a promising solution as application layer protocol, and its specifications open to the implementation and to the use of a different ARQ mechanism, as the use of *NSTART* > 1 would require³. This would allow to take advantage of a large class of devices supporting more complex mechanisms and benefiting of a

²RFC 7252 defines *NSTART* as the number of *simultaneous outstanding interactions* (as in Section 4.7). For the sake of brevity, we refer to it as *transmission window*.

³Section 4.7 of RFC 7252 opens to the possibility of using *NSTART* > 1, if a congestion control mechanism is available.

larger transmission window. In order to reduce the delivery delay of a request-response pattern, like in the case of CoAP, the mechanisms described in the next two paragraphs can be applied.

1) *Observer pattern*: CoAP specifications open to the implementation of the so-called *observer* pattern, which provides a data exchange model semantically close to the PUB/SUB one. A CoAP client performs a registration to the server(s), indicating the URIs it is interested into. Anyway, there is still a difference with respect to the PUB/SUB paradigm: the decoupling between data consumers and producers guaranteed by the publish/subscribe paradigm cannot be provided by the observer pattern. The absence of an intermediate entity leaves the server(s) in charge of keeping a list of the interested clients. The next paragraph explains how the use of a proxy can solve the aforementioned issue.

2) *CoAP proxying*: In order to have a CoAP-based configuration really similar to a PUB/SUB one, a further step is necessary, which exploits the use of the *proxying* functionality, as described in RFC 7252. A proxy is defined as a CoAP endpoint that can be delegated by clients to perform requests on their behalf. Thus, a CoAP proxy is an intermediate entity, which can actually decouple the clients from the servers.

By implementing both the proxying functionality and the observer pattern, as we propose in this work, CoAP behaves similarly to MQTT.

C. Transport protocols

A key difference between CoAP and MQTT is the transport protocol they rely onto. MQTT is TCP-based, which is connection oriented, thus a Three-way Handshake (3WHS) procedure is needed to establish a connection. On the other side, CoAP is UDP-based⁴, which realizes a connectionless communication and does not provide any congestion or flow control algorithms. While the use of TCP can be of interest in some M2M/IoT scenarios [12], [14], the majority of them would largely benefit of a lightweight transport protocol.

In the following section, the performance provided by the use of MQTT and of CoAP are compared, when the latter implements the observer pattern and the proxying functionality.

IV. SCENARIO DESCRIPTION

In this section, the scenario under consideration is described, and the logical architecture is visible in Figure 2. Several CoAP servers produce data that are sent to the CoAP proxies. Each proxy is in charge of delivering the received data to a remote CoAP client via DVB-RCS2-compliant RCSTs. Thus, we assume that the CoAP servers, which encapsulate available resources, act as data producers. For instance, such an architecture can be applied to low-altitude Unmanned Aerial Vehicle (UAV) swarms, whose use is growing more and more common in several application fields, such as the case of precision agriculture [15]. A master UAV acts as a proxy, collecting

⁴A TCP-based CoAP version is in a draft IETF proposal available at <https://datatracker.ietf.org/doc/draft-ietf-core-coap-tcp-tls>

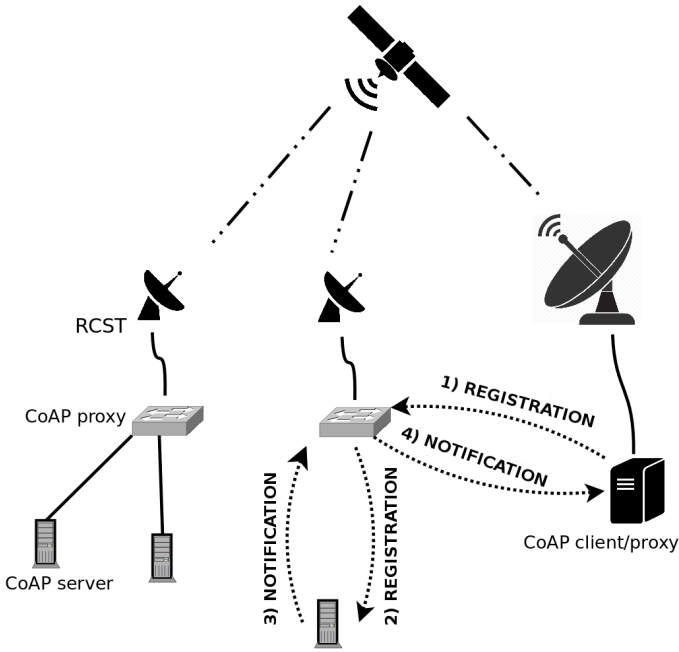


Fig. 2: The CoAP-based scenario under consideration. The dotted lines show a typical setup procedure at application layer (steps 1 and 2) and the notification of new messages via proxy (steps 3 and 4).

data from other UAVs in the same swarm and delivering data via satellite to a remote data center.

We recall that CoAP is UDP-based, thus the ARQ protocol must be implemented at the application layer, if reliable delivery is expected. In Section IV-A, the CoAP settings in use are described, while Section IV-B deepens the description of the system configuration.

A. CoAP protocol implementation

We implemented the CoAP protocol as a Network Simulator 3 (NS-3) module, along with the observer pattern and the proxying functionality. The numerical results in Section V are based on the use of those extensions, thus the typical CoAP request/response paradigm is substituted by a PUB/SUB-like mechanism. The reason behind the latter choice is straightforward: removing the need for a request, the data delivery delay is reduced, because fresh data are available to a client as soon as they are generated or collected by a server. On high-delay links, a *push* strategy can provide large gains with respect to a *pull* one, for instance in terms of delivery delay.

B. System configuration

We consider a network composed by a Geosynchronous (GEO) satellite and a large number of CoAP servers, connected to CoAP proxies; each proxy is connected to a RCST (see Figure 2). It is worth to underline here that a single proxy (instead of multiple ones) can be used if a single network is desired; multiple proxies are to be used if separated networks are required. In Figure 2, we refer to the more

general case with multiple separated networks. CoAP servers produce M2M/IoT-like data that is delivered to a remote CoAP client. If more clients were present on the remote side, a connection per client would be opened, thus increasing the contention level on the random access channel. Alternatively, a receiving proxy can be placed on the remote side, too, in order to have a single connection per sender proxy to the receiving one. Thus, the scenario under consideration can be considered as representative of both aforementioned cases, because the CoAP client in Figure 2 can be substituted by a CoAP proxy, then connected to multiple clients.

In the extensive simulations we ran, data sources begin the transmission according to an exponentially distributed inter-arrival time with parameter λ . The data payload length is randomly drawn, with probability $1/i$, from i Pareto distributions with parameters $x_m^i > 0$ and $\alpha^i > 0$, where $i \in \{1, 2, 3\}$. The three distributions are here meant to represent small, medium and large application M2M/IoT payload lengths, as generated or collected by the server(s). More technically, each CoAP server sends a burst of packets, then forwarded by the proxy, which are packed into a bulk of DVB-RCS2 RA blocks, according to the specifications of Waveform 14⁵, reported in Table I. A single timeslot can be used per RA block by each RCST, according to typical configurations of DVB-RCS2 systems. The MAC protocol in use is Contention Resolution Diversity Slotted ALOHA (CRDSA) [16], configured with 3 replicas. Time is slotted and each RA block is composed of 64 timeslots. The MAC queue length is set to an arbitrary large value and both return and forward links are assumed to be error-free. In the return link, collisions can occur, thus retransmissions are triggered in order to ensure a reliable data delivery. ACKs are assumed to be always correctly received.

V. PERFORMANCE EVALUATION

The scenario presented in the previous section is here numerically evaluated and then compared to the MQTT-based scenario in [3], which is sketched in Figure 3. In this work, the length of the transmission windows of the CoAP servers ranges into $NSTART \in [1, 100]$. Thanks to this, we explore the possibility to reduce the completion time by increasing $NSTART$. In the following, a Go-Back-N [17] ARQ protocol, employing exponential backoff, is in use if $NSTART > 1$. The following numerical results are based on extensive simulation runs based on the use of S-NS3 [18], a satellite network extension to NS-3 platform. The simulator parameters have values as reported in Table I and a minimum of 1000 data exchanges per scenario has been simulated, in order to ensure statistical reliability.

In Figure 4, the completion time of CoAP and MQTT data exchanges is visible, in presence of low/moderate load. The completion time is plotted against an increasing number of application packets sent per data exchange, as readable on the x-axis. Seven increasing $NSTART$ values have been selected

⁵The waveform specifications are drawn from Table A-1 "Reference Waveforms for Linear Modulation Bursts" in [2].

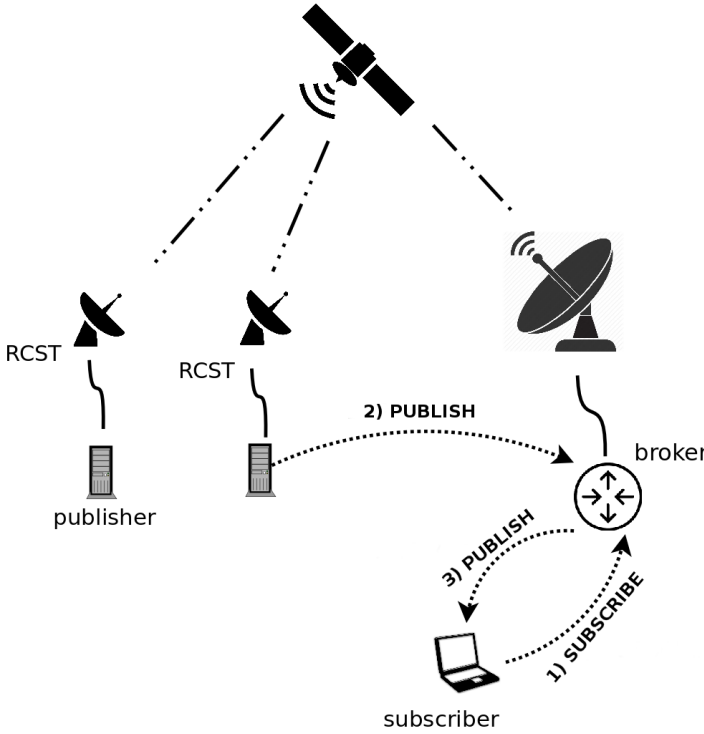


Fig. 3: The MQTT-based scenario in use for comparison. The dotted lines show a typical setup procedure at application layer (step 1) and the notification of new messages via broker (steps 2 and 3).

Name	Value
RA scheme	3-CRDSA
RA blocks per superframe	1
RA block duration	13 [ms]
Timeslots per RA block	64
Gross slot size	188 [B]
Net slot size	182 [B]
Bandwidth	8012820 [Hz]
Roll off	0.2
Carrier spacing	0.3 [Hz]
Nominal RTT	0.52 [s]
Pareto distributions	$x_m^1 = 931$ [B] $x_m^2 = 9532$ [B] $x_m^3 = 47663$ [B] $\alpha^1 = \alpha^2 = \alpha^3 = 1.1$

TABLE I: Simulator setup parameters

Protocol stack	Avg. aggregated goodput
CoAP/UDP ($NSTART = 1$)	32.14 [KB/s]
CoAP/UDP ($NSTART = 2$)	40.46 [KB/s]
CoAP/UDP ($NSTART = 3$)	43.21 [KB/s]
CoAP/UDP ($NSTART = 4$)	45.58 [KB/s]
CoAP/UDP ($NSTART = 5$)	50.1 [KB/s]
CoAP/UDP ($NSTART = 10$)	57.1 [KB/s]
CoAP/UDP ($NSTART = 100$)	77.5 [KB/s]
MQTT/TCP	50.6 [KB/s]

TABLE II: Average aggregated goodput at MQTT broker or at CoAP client/proxy in the scenarios under consideration

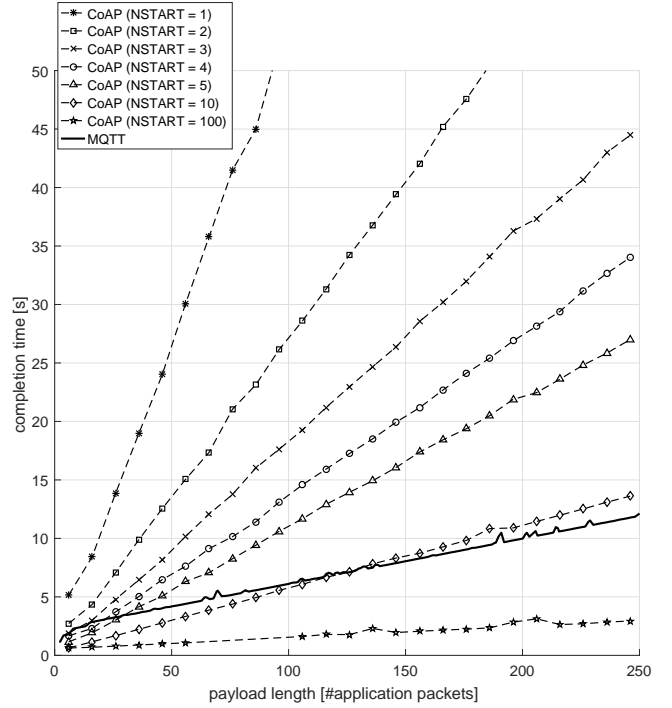


Fig. 4: Completion time of MQTT and CoAP data exchanges per CoAP proxy/client or MQTT publisher.

NSTART	Normalized MAC offered load		
	mean	25th p.	75th p.
1	0,0671	0,031	0,09
2	0,0820	0,047	0,11
3	0,0880	0,047	0,12
4	0,0929	0,048	0,12
5	0,1016	0,048	0,14
10	0,1161	0,062	0,15
100	0,1172	0,063	0,16

TABLE III: Normalized MAC offered load for increasing $NSTART$ values if CoAP is in use at the application layer

in the CoAP-based scenario: the use of a larger value provides a lower completion time, as expected. Anyway, the latter is valid only in presence of a low/medium traffic profile on the RA channel, so that erasures due to collisions are unlikely to occur. Table III reports the normalized MAC offered load for each $NSTART$ value and its 25th and 75th percentiles, when $\lambda^{-1} = 1$ [s]. The collision rate is almost negligible for the load intervals under consideration. A low/medium traffic profile is used, in order to avoid congestion phenomena.

The default CoAP configuration ($NSTART = 1$) provides a quite large completion time, even for small amounts of data, sub-utilizing the available system resources. In case of larger values, the completion time decreases and, for $NSTART = 100$, CoAP provides a lower completion time than MQTT. It is worth underlining here that the Bandwidth-Delay Product (BDP) of the satellite link is ≈ 40 CoAP packets; thus, when $NSTART = 100$ and the burst length is larger than BDP,

backlogging is present.

The completion time of the MQTT-based scenario depends on TCP⁶ congestion control algorithm; as the TCP congestion window increases over time, the curve exhibits an almost linear trend to a first approximation, in presence of a low collision rate on RA channels.

If looking at the completion time, a comparable value is obtained with a CoAP configuration with $NSTART = 10$. Anyway, the different trends in the completion time provided by MQTT and CoAP are clearly visible: in the first part, the completion time in MQTT-based scenario increases faster than in the CoAP-based scenario because of the small TCP congestion window. As soon as TCP congestion window increases because of larger payload lengths, the completion time reduces accordingly.

Table II shows the average aggregated goodput at CoAP client/proxy (see Figure 2), compared with the same at MQTT broker (see Figure 3). Thanks to the lower overhead provided by the CoAP/UDP stack, it outperforms the MQTT/TCP stack. In fact, even if MQTT/TCP is approximately equivalent to using CoAP/UDP with $NSTART = 10$, the larger overhead reduces the achievable goodput w.r.t. the latter configuration, as shown in Table II. Eventually, some considerations are in order: in IoT/M2M scenarios, the use of CoAP can provide some advantages over MQTT, because the length of the transmission window can be set at the application level, as well as the ARQ algorithm in use, thus providing a larger flexibility. Furthermore, the CoAP-based protocol stack exhibits a lower overhead, which is desirable in such application scenarios.

VI. CONCLUSIONS

This work focuses on a comparison between two of the largely used IoT/M2M protocol stacks, based on the use of CoAP and MQTT protocols, implementing the request/response and the PUB/SUB communications paradigm, respectively. The PUB/SUB paradigm can bring large benefits in satellite-based architectures, because of the reduction of the delivery time thanks to the fact that fresh data are sent to registered subscribers as soon they are produced. Because of the latter, in this work, we investigated the use of the CoAP protocol in conjunction with the so-called *observer* pattern and the *proxying* functionality, in order to exploit the advantages provided by the PUB/SUB paradigm, which also provides a fairer comparison than relying on the default CoAP implementation. A qualitative comparison is provided in this work, together with some preliminary numerical results, highlighting how the performance level provided by the use of CoAP outperforms the one provided by MQTT on RA satellite channels, and underlining the flexibility that easily tunable settings at application layer provide w.r.t to lower layer settings. Future work will focus on M2M/IoT communications in presence of higher traffic rates, where the performance provided by CoAP may still need further investigation.

⁶TCP NewReno (RFC 6582) is in use in this scenario.

ACKNOWLEDGMENTS

This work has been partially supported by the Tuscany region in the framework of SCIADRO project (FAR-FAS 2014), and by SatNEx (Satellite Network of Experts) programme, IV phase.

REFERENCES

- [1] S. Tarkoma, *Publish/Subscribe Systems: Design and Principles*. Wiley, 2012.
- [2] "Second generation, DVB-RCS2 Part2: Lower layers for satellite standard," ETSI EN 301 545-2, 2012.
- [3] M. Bacco, T. De Cola, G. Giambene, and A. Gotta, "Advances on elastic traffic via M2M satellite user terminals," in *International Symposium on Wireless Communication Systems (ISWCS)*, Aug. 2015.
- [4] P. Bhave and P. Fines, "System Behavior and Improvements for M2M Devices Using an Experimental Satellite Network," in *Region 10 Symposium (TENSYMP)*, May 2015.
- [5] S. Vassaki, G. T. Pitsiladis, C. Kourogorgas, M. Poulakis, A. D. Panagopoulos, G. Gardikis, and S. Costicoglou, "Satellite-based sensor networks: M2M Sensor communications and connectivity analysis," in *International Conference on Telecommunications and Multimedia (TEMU)*, July 2014.
- [6] L. Durkop, B. Czybik, and J. Jasperneite, "Performance evaluation of M2M protocols over cellular networks in a lab environment," in *18th International Conference on Intelligence in Next Generation Networks (ICIN)*, Feb. 2015.
- [7] N. D. Caro, W. Colitti, C. K. Steenhaut, G. Mangino, and G. Reali, "Comparison of two lightweight protocols for smartphone-based sensing," in *IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*, Nov. 2013.
- [8] M. Collina, M. Bartolucci, A. Vanelli-Coralli, and G. E. Corazza, "Internet of Things Application Layer Protocol Analysis over Error and Delay prone Links," in *7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, Sept. 2014.
- [9] N. Celandroni, F. Davoli, E. Ferro, and A. Gotta, "On elastic traffic via Contention Resolution Diversity Slotted Aloha satellite access," *International Journal of Communication Systems*, vol. 29, no. 3, pp. 522–534, 2016.
- [10] A. Gotta, M. Luglio, and C. Roseti, "A TCP/IP satellite infrastructure for sensing operations in emergency contexts," *Computer Networks*, vol. 60, pp. 147–159, 2014.
- [11] M. Bacco, A. Gotta, C. Roseti, and F. Zampognaro, "A study on TCP error recovery interaction with random access satellite schemes," in *7th Advanced Satellite Multimedia Systems Conference (ASMS/SPSC) (and 13th Signal Processing for Space Communications workshop)*, vol. January, 2014, pp. 405–410.
- [12] M. Bacco, T. De Cola, G. Giambene, and A. Gotta, "M2M Traffic via Random Access Satellite links: Interactions between Transport and MAC Layers," *arXiv preprint arXiv:1609.03387*, 2016.
- [13] M. De Sanctis, E. Cianca, G. Araniti, I. Bisio, and R. Prasad, "Satellite Communications Supporting Internet of Remote Things," *IEEE Internet of Things Journal*, vol. 3, pp. 113–123, 2016.
- [14] M. Bacco, T. De Cola, and A. Gotta, "TCP New Reno over DVB-RCS2 Random Access Links: Performance Analysis and Throughput Estimation," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–6.
- [15] M. Bacco, E. Ferro, and A. Gotta, "UAVs in WSNs for agricultural applications: An analysis of the two-ray radio propagation model," in *SENSORS, 2014 IEEE*. IEEE, 2014, pp. 130–133.
- [16] E. Casini, R. De Gaudenzi, and O. R. Herrero, "Contention Resolution Diversity Slotted ALOHA (CRDSA): An enhanced random access scheme for satellite access packet networks," *Wireless Communications, IEEE Transactions on*, vol. 6, no. 4, pp. 1408–1419, 2007.
- [17] H. Burton and D. Sullivan, "Errors and error control," *Proceedings of the IEEE*, vol. 60, no. 11, pp. 1293–1301, 1972.
- [18] V. Hytönen, B. Herman, J. Puttonen, S. Rantanen, and J. Kurjenniemi, "Satellite Network Emulation with Network Simulator 3," in *Ka and Broadband Communications, Navigation and Earth Observation Conference*, 2014.