proceedings

MDPI

Proceedings

Decentralized P2P Broker for M2M and IoT Applications [†]

Iván Froiz-Míguez [1,2], Paula Fraga-Lamas [1,2,*] and Tiago M. Fernández-Caramés [1,2,*]

[1] Department of Computer Engineering, Faculty of Computer Science, Universidade da Coruña, 15071 A Coruña, Spain; ivan.froiz@udc.es
[2] CITIC Research Center, Universidade da Coruña, 15071 A Coruña, Spain
[*] Correspondence: paula.fraga@udc.es (P.F.-L.); tiago.fernandez@udc.es (T.M.F.-C.); Tel.: +34-981-167-000 (ext. 6051) (P.F.-L.)

[†] Presented at the 3rd XoveTIC Conference, A Coruña, Spain, 8–9 October 2020

Published: 20 August 2020

Abstract: The recent increase in the number of connected IoT devices, as well as the heterogeneity of the environments where they are deployed, has derived into the growth of the complexity of Machine-to-Machine (M2M) communication protocols and technologies. In addition, the hardware used by IoT devices has become more powerful and efficient. Such enhancements have made it possible to implement novel decentralized computing architectures like the ones based on edge computing, which offload part of the central server processing by using multiple distributed low-power nodes. In order to ease the deployment and synchronization of decentralized edge computing nodes, this paper describes an M2M distributed protocol based on Peer-to-Peer (P2P) communications that can be executed on low-power ARM devices. In addition, this paper proposes to make use of brokerless communications by using a distributed publication/subscription protocol. Thanks to the fact that information is stored in a distributed way among the nodes of the swarm and since each node can implement a specific access control system, the proposed system is able to make use of write access mechanisms and encryption for the stored data so that the rest of the nodes cannot access sensitive information. In order to test the feasibility of the proposed approach, a comparison with an Message-Queuing Telemetry Transport (MQTT) based architecture is performed in terms of latency, network consumption and performance.

Keywords: IoT; edge computing; M2M; distributed computing; IPFS; MQTT; P2P

1. Introduction

The growing number of Internet of Things (IoT) devices generates a massive amount of data that has derived into the adoption of different communications paradigms that go beyond traditional client-server schemes. One of such paradigms is edge computing [1], which is based on the distribution of the computing load among different IoT nodes, thus moving part of the processing from the cloud to the edge of the network and then providing lower latency, improved response times and better bandwidth availability. In addition, recent advances on hardware enable creating more powerful and less power-hungry devices. Thus, the latest IoT devices can handle more complex tasks than simple data storage and device-to-device communications.

The mentioned evolution fosters the development of new distributed computing strategies like the one described in this paper. The proposed strategy is completely distributed, in contrast to traditional edge computing approaches, which provide a hybrid environment with distributed computing and a central server (i.e., a cloud). Not delegating information to a central server has become increasingly important, as it is a single point of failure and a potential source of data leaks. Therefore, the proposed solution makes use of a fully distributed Machine-to-Machine (M2M) communications protocol whose performance is compared with Message Queuing Telemetry Transport (MQTT) [2], which is currently one of the most popular M2M protocols.

2. Design and Implementation

Figure 1 shows the proposed communications architecture. In such an architecture a private swarm is a set of peers that belong to the IoT system. Each peer is a device that manages the communications distributed among the different sensor nodes. Every peer provides persistent storage for the data gathered from its edge computing-based network. The communication between a user and the edge computing-based network is carried out within the same Local Area Network (LAN) through a REST API.

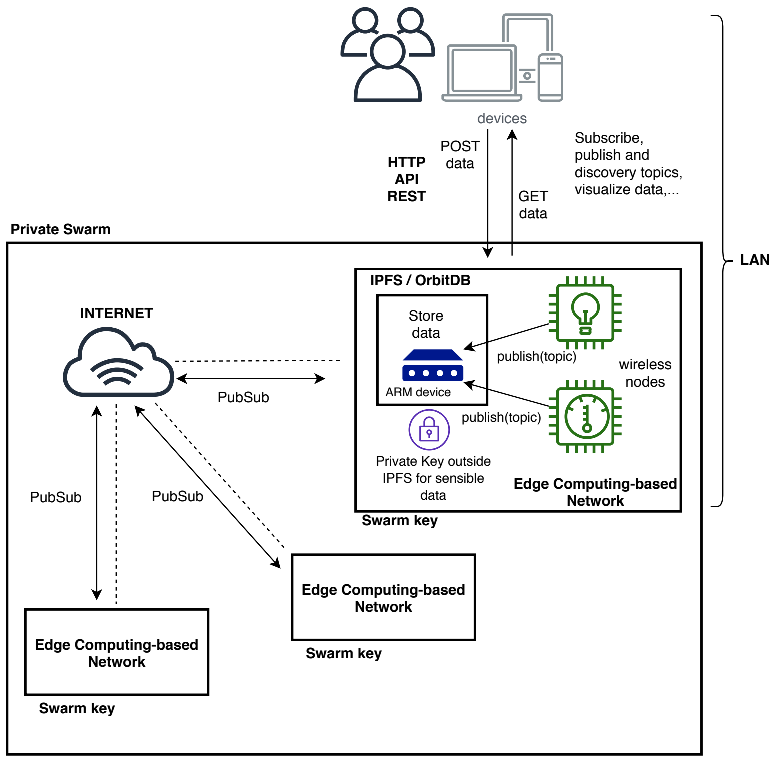
devices
POST data
HTTP API REST
GET data
Subscribe, publish and discovery topics, visualize data,...
LAN
Private Swarm
IPFS / OrbitDB
Store data
ARM device
publish(topic)
wireless nodes
publish(topic)
Private Key outside IPFS for sensible data
Edge Computing-based Network
Swarm key
INTERNET
PubSub
PubSub
PubSub
Edge Computing-based Network
Swarm key
Edge Computing-based Network
Swarm key

Figure 1. Communications architecture of the proposed system.

The devices make use of Inter-Planetary File System (IPFS) [3] to implement a decentralized file system that provides better performance than HTTP when managing large amounts of data. Moreover, the devices use a experimental publication/subscription protocol called PubSub for M2M communications.

For implementing the proposed architecture, a Raspberry Pi was used as a node. It runs a go-ipfs instance with the PubSub function enabled. For persistent storage, OrbitDB (a distributed database that runs on top of IPFS) is used through a port in golang that offers better performance than the original javascript version [4]. In addition, it is possible to communicate with the system via an HTTP REST API to perform actions and get the obtained results.

As edge device, a Raspberry Pi Zero (RPi Zero) was used to receive measurements from different sensors via BLE or WiFi. Thus, the RPi Zero is in charge of the distributed storage and of the M2M communications with other edge devices.

3. Experiments

To determine the performance of the system in terms of latency and throughput, different tests were carried out. The obtained results were compared with the ones provided by an MQTT broker that run in a cloud (an Eclipse Mosquitto broker was deployed in a cloud while a client node (RPi Zero) published messages). In a similar way, an IPFS node hosted in the same cloud acted as a topic subscriber while the client node sent messages. The tests simulated the publication of 10 messages from 10 different clients. The obtained latencies are shown in Table 1.

Table 1. Latency comparison between MQTT (left) and IPFS PubSub (right).

Measure	Latency (ms)	Measure	Latency (ms)
Minimum publish time	41.101	Minimum publish time	320.8
Maximum publish time	69.067	Maximum publish time	375.539
Mean publish time	52.379	Mean publish time	340.272
Standard deviation	5.649	Standard deviation	6.173

In addition, the throughput of OrbitDB was measured by making insertions in an EventLog and then measuring response times. Table 2 shows the obtained results.

Table 2. Performance of an EventLog of OrbitDB.

Number of Insertions	Latency (s)	Throughput (queries/s)
50	7.558	6.615
100	12.777	7.826
500	72.82	6.866

4. Conclusions

The proposed decentralized brokerless system offers a good trade-off between performance, security, and reliability. Although MQTT provides a low latency (maintly beacause PubSub was not designed by having M2M communications in mind), its centralized architecture is prone to security issues that can be easily tackled by the proposed system.

References

1. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, 3, 637–646.
2. Naik, N. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In Proceedings of the 2017 IEEE international systems engineering symposium (ISSE), Vienna, Austria, 11–13 October 2017.
3. Benet, J. IPFS—Content Addressed, Versioned, P2P File System. IPFS White-Paper. *arXiv* **2014**, arXiv:1407.3561.
4. Froiz-Míguez, I.; Fraga-Lamas, P.; Varela-Barbeito, J.; Fernández-Caramés, T.M. LoRaWAN and Blockchain based Safety and HealthMonitoring System for Industry 4.0 Operators. *Proceedings* **2019**, 42, 77.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).