Experimenting with a Fog-computing Architecture for Indoor Navigation

Pietro Battistoni, Monica Sebillo and Giuliana Vitiello
Department of Computer Science
University of Salerno
Fisciano (Salerno), Italy
Email: {pbattistoni,msebillo,gvitiello}@unisa.it

***Abstract* — The present paper outlines the framework design of an Application Agnostic Architecture for Fog and Mist Computing systems. An analysis explains how the proposed architecture satisfies the primary characteristics of a Fog and Mist computing paradigm by using low power and low-cost components. It also introduces an Indoor Navigation System and a Notification service as a use-case, to assess the proposed framework as a Fog-Computing architecture while demonstrating how it can address Mobile and IoT challenges.**

***Index Terms* — Fog Computing, Mist Computing, Edge Computing, MQTT, Indoor Navigation System, ESP32.**

I. Introduction

DBS Group research, in [1], defines the Internet of Things (IoT), as the pillar of Artificial Intelligence (AI) and, although in 2018 the IoT sets only for the 14% the path for developments in AI, its adoption will rise to 176% in 2030 with primary uses that will embrace Consumer, Manufacturing, Utilities & Energy, Transportation & Logistics and Telecommunication sectors.

In [2] Cisco Internet Business Solutions Group estimates that the interconnected devices will reach 50 billion units by 2020 with solutions that need analytics and low latency answers. Such a continuous increasing of connected devices with their massive amount of fast data produced requires new concepts and new technologies to turn the growing fleet of IoT and Mobile devices into serviceable data sources. A Cloud architecture, although satisfying some of the requirements, cannot address all the emerging challenges. Indeed, a Centralised Cloud service represents a severe single point of failure when the Internet connection is not guaranteed, and the continuously increasing number of connected devices represents a challenge for the connection bandwidth. In most cases, the optimisation concerning costs for mobile and wide area connectivity is a matter. Many of connected devices need to take prompt action, based on events or data analysis produced by other physically close devices, and in these cases, the low transmission latency is a precise requirement that *Cloud* computing usually cannot satisfy.

A *Fog-Computing* architecture can represent a solution. It resides in multiple layers between the connected device and *Cloud* services and represents an extension of traditional *cloud-only* models and not an alternative (Fig. 1).

This paper proposes the design of an Application Agnostic Architecture framework for *Fog-Mist* Computing systems, where.

Finally, an *Indoor Navigation Support* application, with a messages based notification system, reveals the proposed *Framework* as a proper *Fog-Computing* architecture.

The paper is organised as follows. In Section II a brief description of *Fog* and *Mist* Computing paradigms are given.

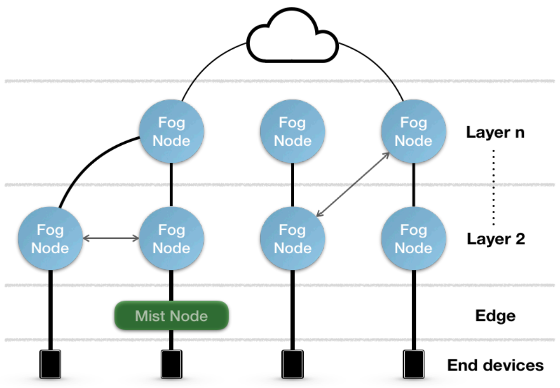

Fig. 1. Fog Computing Architecture

Section III describes the architecture of the proposed *Framework* and details its components, while Section IV presents a use-case based on this *Framework* as proof of concept. Section V argues the *framework* as a *Fog-Computing* system. Finally, Section VII concludes the paper and introduces some future works.

II. Preliminaries

A. Fog-Computing

The NIST Fog Computing Model [5] defines *Fog-Computing* as a layered model that facilitates the deployment of distributed applications and services, enables low latency, reliable operations, and removes the requirement for persistent cloud connectivity. The architecture of a *Fog-Computing* solution consists of *Fog-Nodes* placed between the smart end-device and centralised *cloud* services. *Fog-nodes* give local computing resources, storage and network connectivity to the end-device. Moreover, this model aims to offer ubiquitous access to a shared continuum of scalable computing.

B. Fog-Node

One or more connected *Fog-Nodes* deploy *Fog-Computing* services. A *Fog-Node* has to provide the smart end-devices at the edge with some form of data processing, computing resource and communication services between network layers. In some cases, *Fog-Nodes* may also offer data storage. A *Fog-Node* operates in both centralised and decentralised way and can be configured as stand-alone, communicating with other nodes to deliver services, or can be federated to form clusters, providing horizontal scalability over diffuse geolocations. *Fog-Nodes* are geographically distributed and have to be aware of both their geographical position and logical location within the context of their cluster.

C. Mist Computing

The NIST Fog Computing Model [5], defines a *Mist-Computing* as a simpler form than *Fog-Computing*, where the

This research is partially funded by the E3APP project, POR CAMPANIA FESR 2014/2020

Mist-Nodes are more specialised with lower computational resources. They are placed directly within the edge of the network fabric, closer to the end-device than the more powerful fog node they feed. Often, they share the same locality with the smart end-device to which they give connectivity and computational services with lower-latency.

D. *Fog-Computing is not Edge Computing*

As bringing elaboration closer to the edge is also the goal of *Edge-computing,* there are significant differences between it and the *Fog-Computing* concept.

Edge computing executes specific applications in a fixed logic location, usually an appliance, and provides just a direct connection to the *Cloud* services. Instead, the *Fog-Computing* runs applications in a multi-layer architecture allowing dynamic reconfigurations and performing computation, storage and transmission services. The *Fog-Computing* can be thought as a continuous data transformation into knowledge. At each layer, the data analysis decreases the data quantity and increase their information, while from the edge they are reaching the centralised *Cloud*. Furthermore, this information can be accessed at any layer by other *Fog-Nodes*, thus reducing the path and latency between data producer and data users, when it is required.

III. The Proposed Framework

This Section describes the proposed framework in terms of layer composition and selected devices.

A. *The Layered Model*

Fig. 2 depicts the proposed layered model, numbering the layers from bottom to up.

Layer 1 consists of partially meshed hardware devices at the very edge of the network. *Layer 1* devices connect the *user's mobile device* with the device in *Layer 2*, downstream and upstream respectively, while a partially meshed network chains all devices in *Layer 1*.

Layer 1 is a *Mist Layer*, because of the functionality and the computational capacity of its nodes.

The device in *Layer 2* connects devices of *Layer 1* in downstream with the Internet in upstream, reaching the Cloud. *Layer 2* is a *Fog-Layer*

B. *Layer 1 devices*

The network topology is made up of several *mist nodes*. One of the *Mist-Nodes* in Layer 1 is defined as *Root Mist-node* and connects its Wi-Fi Station Interface (STA) to the Wi-Fi Access Point in *Layer 2* (Fig. 3). The STA interface of subsequent *Mist-Node* in *Layer 1* connects to the Soft AP interface of the *root Mist-Node*. The Soft AP interface of the remaining *Mist-Nodes* in *Layer 1* accepts connections from STA interface of subsequent *mist node* and from the below nearest user's *mobile devices* if any. Then, each *Mist-Node* can transmit its packets to previous *Mist-Node* and at the same time serves as a relay for the next node and the *user-devices*, as depicted in Fig. 3.

At *Layer 1*, *ESP32-WROVER* module-based devices are deployed. This module made by the Espressif Systems has the Wi-Fi and Bluetooth Light Energy (BLE) coexistence. The ESP32 is highly programmable and low cost (less than 4$ in a single piece) System on Chip (SoC). It integrates many interfaces in a single chip and consumes very low power. The "ESP32 Technical Reference Manual" [3] gives more details. Despite its two Xtensa® 32-bit LX6 Microprocessors and the Ultra-Low Power co-processor (ULP), it is a right candidate for the *Mist-Nodes* of the proposed model.

The Espressif IoT Development Framework, based on ESP-IDF Software Development Kit (SDK) [4], offers a good set of APIs to deploy on the SoC the firmware for the mesh network (ESP-MESH), and BLE functionalities.

The partially meshed network of *Mist-Nodes* can have an extensible coverage area, as only one of *Mist-Nodes* is required to connect to the central *AP*. The Wi-Fi Network is also less susceptible to overloading, as a single *AP* no longer limits the number of end-user mobile devices allowed on the network.

The end-user mobile devices are defined as *Leaf-Node* since they cannot have any child nodes, which means no downstream connections. They can only transmit or receive their packets. In the proposed framework, the *Leaf-Nodes* join the network, connecting to the *Mist-Node*, which advertises the strongest Received Signal Strength Indication (RSSI) from its *SoftAP* interface.

The *Root Node* is the *Mist-Node* designed as Root during configuration, or dynamically elected, based on the signal

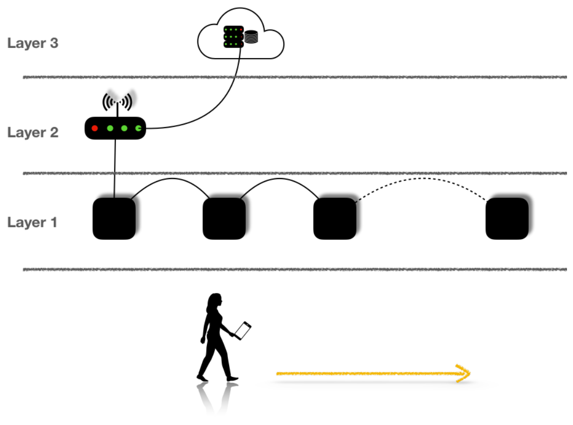

Fig. 2. The proposed Framework scheme

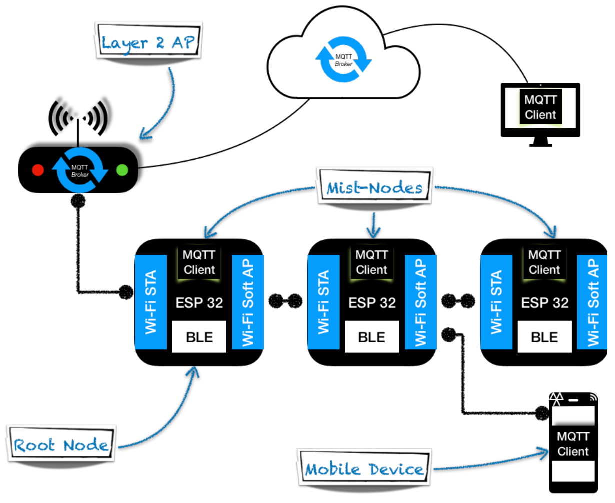

Fig. 3. The partially Meshed Network

strength estimated between each node and the *Fog-Node* of *Layer 2*.

C. Multi-hop network

Layer 1 is a multi-hop network. Each *Mist-Node* device starts in an idle state, listening for Wi-Fi beacon frames. It generates a list of potential parents and forms an upstream with the strongest *RSSI*, moving out from the idle state. Building the mesh network can be autonomous and flexible, where *Mist-Nodes* can dynamically leave or join the network.

D. Layer 2 device

At *Layer 2* the Raspberry Pi 3 Model B+ is deployed as a *Fog-Node.* This device has an ARMV8 64-bit SoC @ 1.4GHz, Wi-Fi, Bluetooth, BLE, Gigabit Ethernet connectivity, and supports Micro SD cards to store operating system and data, for a cost of less than 50$. The officially supported operating system, the *Raspbian,* comes with plenty of software for programming and general purposes. All typical functionality of a Wi-Fi Access Point along with the *Mosquitto* MQTT Broker is deployed on it.

IV. A Scenario of Indoor Navigation

The described *framework* is used to design an *Indoor Navigation System* (INS), with a *mobile end-user* notification system, allowing users interaction among them, and with a centralized system. In order to realize the INS, it is mandatory to know the user's position in real-time. To reach this goal, an Indoor Positioning System (IPS) is proposed, where BLE *beacons* [8], transmitted from *Layer 1* nodes, are used on a proximity-based strategy, allowing the end-user application to know its own position on a map and suggest the convenient path to reach the destination.

In order to realize the notification system, an *MQ Telemetry Transport* (MQTT) protocol solution is assumed. It is a lightweight *publish/subscribe* (Pub/Sub) messaging protocol, where multiple clients connect to a broker and subscribe or publish to topics they are interested in.

Introduced in 1999 by Dr Andy Stanford-Clark from IBM, and Arlen Nipper from Arcom, for constrained devices and low-bandwidth, high-latency or unreliable networks, MQTT is widely used in emerging *Machine-to-Machine* (M2M) and IoT, which need of an extremely simple messaging protocol, and a small footprint client code, when battery power and bandwidth are an issue. Many clients may subscribe and publish to the same *topics* and share messages, asynchronously. This characteristic makes it a good candidate even for a chat and message-based notification system as in the case of Facebook® Messenger application [9]. The MQTT solution is built on a *Client-Server* architecture, with a so light client that can stay even on microcontroller devices with small memory and processing capacity. A basic solution is made of a *Broker* (server side) and many clients. *Clients* should connect to a *Broker* and *Subscribe* to a *Topic* they are interested in. Any client connected to a *Broker* can *Publish* to a *Topic* and the *Broker* routes the message to all *clients*, which subscribed to the just published *Topic*. In such a way, clients can exchange messages in an asynchronous mode. Furthermore, a *Broker* can be bridged to another *Broker*, and the *clients* can still exchange messages, connecting to any of bridged *Brokers*.

In this project, the open source "Mosquitto" is used as MQTT broker [10] (server part) and the *client* part is developed by using Paho libraries [11]. Both software are an Eclipse™ project. With a *Broker* deployed on the device in *Layer 2*, any clients deployed on each user's mobile device, can Pub/Sub to local *Broker*, by using the Wi-Fi access from any of the node in *Layer 1* (Fig. 3). As the MQTT broker on *Layer 2* and the end-user mobile devices are on the same WLAN, the message latency is low, and messaging service is guaranteed even when the Internet connection is lost. At the same time, as the *Broker* on *Layer 2* will be bridged with another *Broker* on the *Cloud*, Internet *clients* can join the notification-messaging system, directly connecting to the cloud *Broker*.

A. Bluetooth Light Energy (BLE)

The work in [12] presents four Generic Access Profile (GAP) roles for a BLE device. Only the *Broadcaster* and *Observer* GAP are used for the proximity location solution. A *Broadcaster* is a device that sends advertising packets, so it can be discovered by the *Observers*. This device can advertise but cannot be connected. An *Observer* is a device that scans for *Broadcasters* and reports this information to an application. This device can only send scan requests but cannot be connected.

The *Mist-Nodes* of the proposed *framework* acts as *Broadcasters* and their BLE interfaces are sets as Scannable Undirect Mode, i.e. the node can be discovered by any other device but cannot get connected to it.

The end-user device acts as an *Observer* and *forwards* the *Broadcaster* information to the mobile application, particularly the Bluetooth RSSI, which will give the estimation of the distance between the mobile device and the *Mist-Node*. This last information will be used to localize the mobile device along the *Mist-Nodes* path.

B. Messaging

The MQTT protocol is used for many tasks. It is used to pass parameters among the nodes, notification and short messages among the user-devices, and for centralized Orchestration of nodes. The Topic is crucial for message routing among MQTT *clients*. In the present *use-case*, the *end-user mobile devices*, *Mist-Nodes*, and eventually the Web applications that allow centralized management of the system, are MQTT *clients*, which *publish* and *subscribe* on *Topics*. It is not necessary any configuration about *Topics* to create it, it is enough to *Publish* or to *Subscribe* to it. Briefly, the *Topics* are a hierarchical structure, which uses a slash ('/') as a separator and the characters '+' and '#' as a wildcard.

When a *client* publishes a message "arrived" on a *Topic* "Node1/location/", and a message "call me" on a *Topic* '/Node1/msg/' the *Topics* are created. Any client that has subscribed to '/Node1/msg/' will receive the message "call me", while the clients which subscribed to '/Node1/#' will receive the messages "arrived" and "call me". The definitions of *Topics* are strictly application related. According to the example of Fig. 4, the following scenario can be figured. A mobile application deployed on a user device, after joined the WLAN network through the *Mist-nodes*, can publish the message "arrived" on Topic 'London-SSID/Client-ID/msg/'. The clients that subscribed to 'London-SSID/Client-ID/msg/' will receive the message as notification of the presence of Client-ID at London-SSID, in real time.

Another interesting mechanism is the bridging among the *Brokers*. When one *Broker* is bridged to another one, it is possible to set that all messages of a *Topic*, or a subset of *Topics*, published on any one of the *Brokers*, will be published on the others, too. *Clients*, which subscribe to a specific *Topic*, whenever they are connected to one *Broker* or to the other one, will receive the messages. As displayed in Fig. 4, the

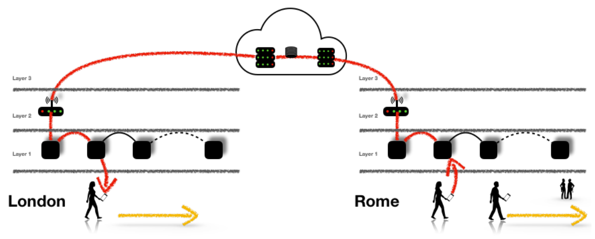

Fig. 4. Multi-Branch company example

notification of the presence of the Client-ID device, will be received by any *Client* connected to the *Broker* in London or to the *Broker* in *Rome*

V. Is the Proposed Framework adherent to the Fog-computing?

In this Section, an assessment of the proposed framework as a *Fog-Computing* system is carried out, according to the criteria presented in [5], [6] and [7].

A. *Low latency*

The proposed framework supports rich services at the edge of the network, satisfying applications with low latency requirements. As *Fog-Nodes* are co-located with the user's *smart devices*, data generated by these devices are processed much quicker than by a centralized cloud service.

B. *Geographical distribution, Interoperability & Federation*

Guaranteeing a widely distributed and geographically-identifiable deployments is one of the most important characteristics of a *Fog-Computing* paradigm, in sharp contrast to the more centralized cloud service. The *Fog-Nodes* of the proposed *framework* acquire their geographical position and are discoverable by it. To better explain the potentiality of this architecture, let geographically scale up the scheme of Fig. 2, as in a multi-branch company (Fig. 4).

Personnel in the Rome office would like to be notified when the company's accountant, reaches the London office. The app for personnel in Rome, Subscribes to a topic like "London/accountant". When the accountant reaches the office in London, her application will publish any message to the topic "London/accountant" and the *Fog-Nodes* in *Layers 1* (both in London and in Rome) will automatically route the new message to all subscribers of the topic, and they will be notified. The client applications need only to know the address of their local Broker, and this is autonomously known because it is the same of local *Fog-Node* at *Layer 1* which, in turn, is known to the *Root Fog-Node* (which is directly connected to it) and propagated to the mobile devices through the *Mist-Nodes* used for the Wi-Fi network access.

C. *Heterogeneity*

Fog-Computing supports collection and processing of data from different form factors, acquired through multiple types of network communication capabilities. The nodes are heterogeneous in term of processing power and storage capacity. The proposed framework adheres to these criteria, as each node has connection interfaces supporting multi-protocol communications and data collection. Both at *Layer 1* and *Layer 2*, the nodes support the *Bluetooth* and *Wi-Fi* for network connection, *Inter Integrated Circuit* (I2C) *and Serial Peripheral Interface* (SPI) for serial communication with sensors and devices. Moreover, the *Fog-Node* at *Layer 2* has also wired Ethernet connectivity. The proposed *Mist-nodes* and *fog-nodes* have different computational and storage capacity, empowering the *Fog-Computing* heterogeneity.

D. *Real-time interaction*

Fog-Computing applications are more appropriate for real-time elaboration rather than batch processing. The proposed framework mainly uses the *MQTT* protocol, which even if it can work in an asynchronous way, it is essentially a real-time message queued protocol and the whole architecture is modelled to achieve low latency at the edge, allowing real-time reaction.

E. *Scalability and agility of federated, fog-node clusters*

The model is easily vertically scalable by adding more *Fog-node* layers, when additional intermediate levels of processing are required, and horizontally by adding more interconnected nodes on the same layer, both locally and remotely placed as in Fig. 4.

F. *Security*

A secure environment requires two features called, *Secure Boot* and *Flash Encryption*, both present in the proposed nodes.

Secure boot represents the technology addressed to assure that only the genuine code is running on the device chip. In the proposed nodes, the code loaded from a flash memory, is verified at each reset.

Flash Encryption consists of encrypting the contents of flash memory so that, physical readout of it is not enough to recover the most of data.

G. *Manageability and Orchestration*

Manageability and Orchestration represent other characteristics that the mist and *Fog-Nodes* may have. They allow the centralized inventory and setup of deployed nodes, as well as the firmware update when required. To accomplish this functionality, the use of *MQTT* protocol is leveraged in the proposed solution.

VI. Conclusion and Future Works

In this paper, a framework based on **low-cost** components has been discussed, which satisfies all the primary characteristics of a *Fog-computing architecture*. It can be deployed to many scenarios where a *Fog-computing* solution is the most appropriate. It can also be scaled up, fine-grained, to satisfy wide geographically distributed solution when most of IoT devices produce data about strictly territory-bond phenomena, which has to be fast treated, close to the user location, and in a timely manner, as in the indoor navigation or in more extensive cases as in [13], and [14]. The proposed *framework* as used in the described *use-case*, by federating the low latency edge-processing with the centralised Cloud computing resources, perfectly adhere to the new fog-computing paradigm.

As future work, an experimental prototype can be developed to test the framework within further real scenarios, where indoor location and geo-analytics can benefit significantly.

REFERENCES

[1] DBS Group research, "Internet of Things The Pillar of Artificial Intelligence," DBS Group research, 2018.

[2] D. Evans, "The Internet of Things How the Next Evolution of the Internet Is Changing Everything," Cisco Internet Business Solutions Group, 2011.

[3] Espressif Systems, "ESP32 Technical Reference Manual," 2018. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf. [Accessed 16 February 2019].

[4] Espressif Systems, "API Reference," [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/index.html. [Accessed 15 02 2019].

[5] National Institute of Standards and Technology, Fog Computing Conceptual Model, National Institute of Standards and Technology, 2017.

[6] OpenFog Consortium, "OpenFog Reference Architecture for Fog Computing," OpenFog Reference Architecture for Fog Computing, 2017.

[7] D Carla Mouradian, Diala Naboulsi, Sami Yangui, Sami Yangui, Monique J. Morrow, Paul A. Polakos, "A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges," 2017.

[8] Google Developers, "Get Started with Beacons," Google, 25 May 2018. [Online]. Available: https://developers.google.com/beacons/get-started. [Accessed 15 February 2019].

[9] L. Zhang, "Building Facebook Messenger," Facebook, 12 August 2011. [Online]. Available: https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920. [Accessed 15 February 2019].

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

[11] iot.eclipse.org project, "Python Client - documentation," [Online]. Available: https://www.eclipse.org/paho/clients/python/docs/. [Accessed 14 2 2019].

[12] Espressif Inc., "ESP32 Bluetooth Architecture," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_bluetooth_architecture_en.pdf. [Accessed 2018].

[13] M. Sebillo, M. Tucci, G. Tortora, G. Vitiello, A. Ginige, P. Di Giovanni, "Combining personal diaries with territorial intelligence to empower diabetic patients" in Journal of Visual Languages and Computing, Elsevier. Volume 29, August 2015, Pages 1–14. DOI 10.1016/j.jvlc.2015.03.002

[14] A. Ginige, L. Paolino, M. Romano, M. Sebillo, G. Tortora, G. Vitiello, "Information Sharing among Disaster Responders - An Interactive Spreadsheet-Based Collaboration Approach", in Computer Supported Cooperative Work (CSCW), vol. 23 (4-6), 2014, pp. 547-583. ISSN: 0925-9724