Event-Driven Cooperative-Based Internet-of-Things (IoT) System

M. Rescati[1], M. De Matteis[1], M. Paganoni[1], D. Pau[2], R. Schettini[1], A. Baschirotto[1], *Fellow, IEEE*
[1]University of Milano Bicocca, Italy.
[2]Advanced System Technology, STMicroelectronics, Italy

Abstract— **This paper presents the development of an Internet-of-Things (IoT) Cooperative System (IoT-CS) based on local Event-Driven response. After introducing IoT basic concepts and most common computational schemes (with particular attention on the differences between Cloud, Fog, and Cooperative computing paradigms), an innovative scheme for IoT will be presented. Thanks to the introduction and development of several intermediate layers between the sensor network and the Cloud, the hereby proposed system allows overcoming most of the problems of the state-of-the-art paradigms. In order to validate the proposed solution the most relevant aspects of this implementations will be presented.**

Keywords—**Cloud computing, Cooperative computing, Edge Device, Event-Driven, IoT, Fog Computing, IoT Healthcare.**

I. Introduction

Internet of Thing (IoT) is one of the most rapidly growing and innovative technologies in recent years, due to the wide and deep applications it covers. Some examples are wearables, healthcare, smart industrial manufacturing, smart cities, agriculture 2.0, datacenters and autonomous vehicles [1]. This means that IoT could potential change our vision of the world and our way of living in just few years.

An IoT system is generally composed by 4 main elements: sensing, communication, computation, and actuation, as shown in Fig. 1. Sensing elements have been greatly improved [2], reducing costs and increasing sensors number and variety. Moreover, communication has been improved, thanks to advanced enabling technologies such as BLE (Bluetooth Low Energy).

This paper deals with the computation and the actuation aspects, i.e. on the ability of the IoT network of elaborating the acquired information and of producing with improved timing constrains output actuation signals as consequences of the read capabilities. The part related to the specific algorithms to be implemented in each computation sites is not here addressed, since it is strongly dependent on the application.

Most of IoT systems ([3], [4], [5], [6]) are based on Cloud-Computing (CLC) ([7]). In CLC systems all data are sent to the Cloud where all the computation and decision-making tasks are executed. This scheme is beneficial because it exploits the high computational and memory power of the Cloud. In addition, this scheme allows very efficient implementation, and relatively low maintenance costs. Cloud-based system is ideal for computation intense application and data-driven applications.

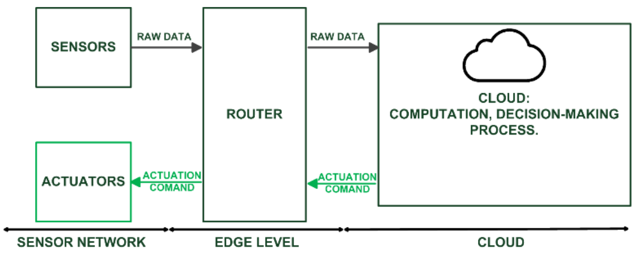

Fig. 1 – Current cloud-centric computing scheme

Nonetheless, CLC scheme presents important drawbacks, as:
- low scalability, due to limited number of sensors (due to the serial Cloud communication);
- few types of connectable device (generally routers cannot connect all type of devices to the Cloud);
- high unpredictable latency time, due to the two transmissions to and from the Cloud-server from reading to actuation;
- privacy issues due to adopted standard, mainly selected in terms of low-power consumption of the nodes;
- big data transfer from the nodes to the Cloud and viceversa
- operation only in zones with internet Cloud coverage.

To overcome these Quality-of-Service (QoS) limitations, the Fog (FC) and the Cooperative (CoC) computing paradigms are proposed ([8] [9] [10] [11]), since recent technological developments increased the possibility of performing computational tasks in local and mobile systems. Thus, FC distributes computational operations among all edge devices (Fig. 2), overcoming most CLC problems. In fact, latency time is strongly reduced [12], since local tasks do not require constant communication with the Cloud. However, FC systems presents some drawbacks, as:
- hardware blocks complexity (FC schemes require more devices to do computation);
- power consumption (more complex blocks consumes higher power);
- limited computing power of the Edge devices.

To overcome both CLC and FC limitations, the CoC paradigm [13] was proposed (see Fig.3), whose main concept is optimizing the computational power, the power consumption and the amount of communication data by distributing computation tasks in each node of the IoT network. This scheme is particularly powerful since it enables optimization of any target system performance. For example, in applications requiring a fast response, system latency is minimized by means of local computation, whereas this is not possible with CLC based systems without reducing the global system speed.

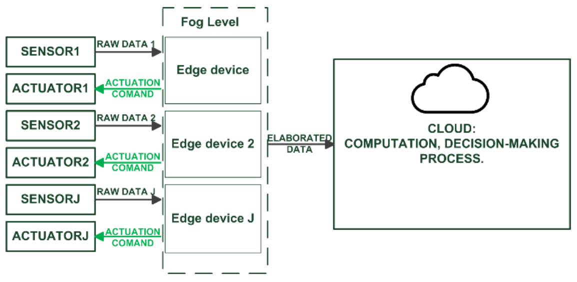

Fig. 2 – Fog computing scheme

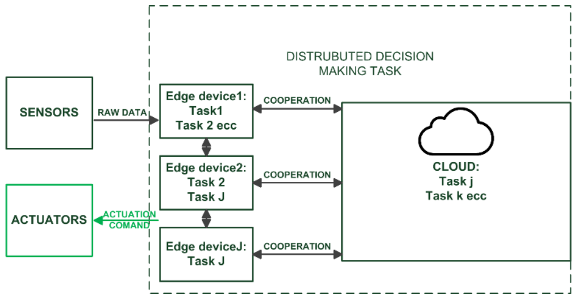

Fig. 3 – Cooperative computing scheme

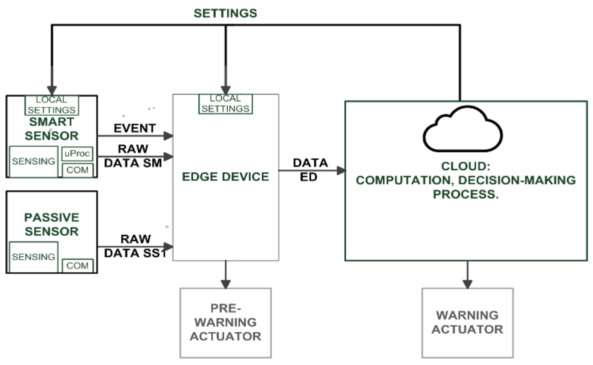

Fig. 4 – Proposed system structure

The main drawback of both CoC and FC are the difficulties in realizing and implementing an efficient system that actually works and is reliable and robust. Another key aspect is the lack of customization in local networks, and the difficulties to locally patch the system [14]. This is one of the reasons because most existing IoT systems are CLC-oriented.

This paper is focused on investigating an *Event-driven* CoC architecture for advanced sensor networks, with the main aim of improving system-level power efficiency and reducing the amount of data to be sent to the Cloud and there processed. In order to validate the architecture idea, a specific system composed by a single Edge device and a sensors network, has been implemented and validated by behavioral simulations and preliminary actual implementation.

II. PROPOSED SYSTEM STRUCTURE

The IoT Cooperative-Based system (IoT-CS) presents new interesting features, like the local distributed computation. This improves the efficiency of the latency approach, reduces global power consumption and increases the computational load, w.r.t. the state-of-the-art paradigms. The system elements are:

- The *Sensor Network,* in turn composed by a certain number sensors node. There are two different sensor typologies used in the network:
  - *Passive Sensors:* devices without computing power (i.e. w/o programmable microcontroller) delivering only raw data;
  - *Smart Sensors:* devices with some local computing power, where specific firmware can be implemented to deliver, alongside with the raw data, also an *Event Signal*, produced by dedicated algorithms executed in the local computing unit (based on Neural Network and/or other methods) and based on local settings;
- *Raw Data*: the data that each sensor of the network acquire and send as they are, without any elaboration;
- *Event signals:* additional signals generated by the execution of dedicated algorithms on the raw data in the local computing unit.
- *Edge devices*: double role devices: routing the raw data from Sensor Network to the Cloud, and acting like a smart intermediary between the Cloud and the sensors. It reacts to the Event-signal produced by smart sensing and actuates a rapid response (much faster than the one from Cloud) and support the smart sensors in their computational task.
- *Cloud-based Server*: in traditional systems, the server performs the heavy computational work, due to its superior computational power and its large data storage capability. In this IoT-CS topology the Cloud provides local settings to enable a fast and accurate local response from local devices.

The architecture of the hereby proposed IoT-CS is then shown in Fig. 4, where the system operates to provide two kinds of response: a pre-warning response (at local level, hence rapid and not so accurate) and a warning response (at Cloud level, hence slower but very precise).

A. Operating Modes

In consideration of the trade-off between power consumption and system performance, the proposed system operates in two different modes:

- *High Performance Mode (HPM)*: the devices in IoT network are always fully working, gathering as much information are possible and passing them to the Cloud, with the produced Event signals. The full operation of all devices requires high power consumption, an important system drawback, since sensors are usually battery powered.
- *Low Power Mode (LPM)*: heavily focuses on *Event-driven* reaction (see Fig. 5). The ED is typically in standby and consumes very little power (*idle state*). When an Event-signal is produced by one Smart-Sensor, the ED activates (*active state*) and starts gathering as much information as possible from the full network, i.e. from all the other sensors. The system remains full working until local (ED) or server calculation decides the Event completion and drives the ED to the idle state. In this operation mode only Event-related data are transmitted to the Cloud and then the data stored in the Cloud are not formally referred to 'standard' operation that allows defining the local settings. For this reason, in LPM, the full operation requires a training phase in HPM for general and local setting definition. The two modes can be alternatively active when required (i.e. during the night could be used the LPM and during the day the HPM).

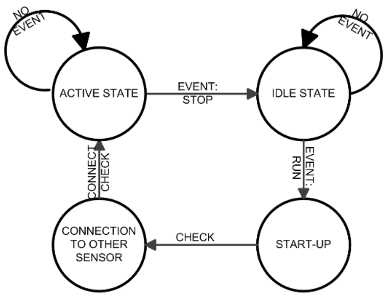

Fig. 5 – State diagram for LPM

B. System Features

Based on these operation modes, the required features are:

• *Event-driven ED activation (from Smart-sensor)*: essential in order to activate the ED when in idle-mode during LPM cycle.

• *Gateway-Smart Sensor communication:* the system must be able to read, translate and manage different kind of signal locally (*Event data* and *raw data*).

• *Sensors data request:* in case the default data-rate from the sensors does not met the event-driven operations requirements

• *Local pre-warning check:* performed by local computing power in the Smart Sensor or in the ED, evaluated by the local firmware operating using sensors data and local settings.

• *Server communication:* implementing a communication standard that handle: data routing to the server, settings (i.e. threshold) from the server, actuation signals from the server.

• *Server global warning check:* generated based on the big-data available on the server (relative to the history of all the sensors during the training and/or during the Event response) done using the higher computational power in the server.

• *Local settings update:* generated by the server in order to align (low-volume) local settings to the (large volume) Cloud data and analysis.

III. Home IoT: A case study

All of the above considerations and features are hardware implemented in the project "Home IoT", whose final goal is the realization of a Smart HealthCare IoT system, with particular attention to elderly assistance at home and in the car.

A prototype hardware of the IoT-CS has been implemented to validate the previous concepts. It is composed by a large Sensor Network (SN), one single Edge-Device (ED) and the Cloud. (all present in the photo in Fig 6.) The most important hardware characteristics are:

- *Sensor Network.*
  - *Simple Sensors for Health Monitoring.* two simple sensors monitor the human being activity: a BLE Hearth Rate wristband, and a proximity sensor (X-NUCLEO-6180XA1), connected via USB serial to the gateway.
  - *Smart sensor*: SensorTile, a board (from STM), equipped with: humidity, barometer and temperature sensors; microphone; 3D accelerometer; 3D gyroscope; 3D magnetoscope; microprocessor (ARM cortex M4); a BLE communication module. Hardware and software are fully customizable, due to the design of the expansion cradle and full API and middleware support [15];
- *Edge Device:* the adopted ED is the Board ST B2260 that is equipped with all the connectivity required for the system (Bluetooth 4.0, Wi-fi, Serial port input, Ethernet port, HDMI video output) and enough computing power for the application (including local calculation) due to an ARM-based dual processor (ARM Cortex-A9) [16].
- *The Cloud-Server* The Cloud-server operations are implemented in a x86 Laptop, which is capable of the required storage and computational power, for this use case.

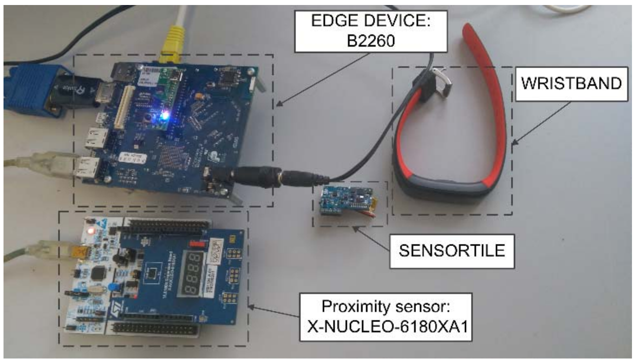

Fig. 6 – Hardware used for the proposed demo

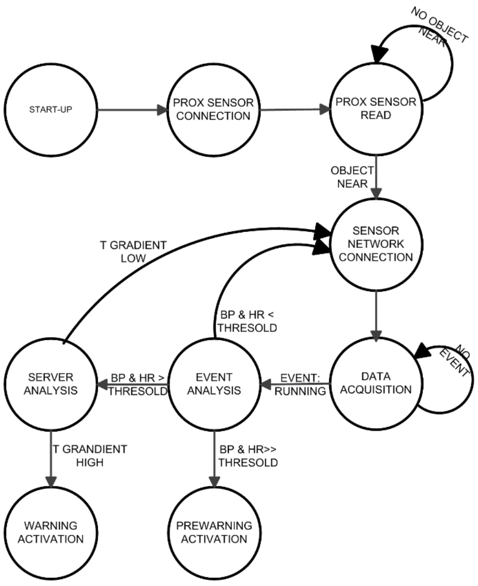

Fig. 7 – State diagram for the use case demo

Based on the operations described in the second paragraph a demo of an IoT-CS has been set, implementing a use case, able to monitor the activity of a person and identify dangerous situations. The state diagram of the implemented use case demo is represented in Fig. 7.

Parameter	Data rate	Dynamic Range	Resolution
Temperature	1 Hz	-40/120 °C	32 bit
3D Acceleration	10 Hz	-16g/+16g	3x16 bit
3D Gyroscope	10 Hz	-2000/+2000 dps	3x16 bit
3D Magnetoscope	10 Hz	-50/+50 gauss	3x16 bit
Pressure	1 Hz	260-1260 hPa	16 bit
Humidity	1 Hz	0/100% r.h.r.	16 bit
Noise level	1 Hz	0/120 dBSPL	16 bit
Heart Rate	Every 20 min	0-180 HBPS	16 bit
Blood Pressure	Every 20 min	40-200 mmHg	32 bit

Table 1 - Parameters and data rates in HPM

When the system is booted, in order to minimize the power consumption, only starts acquire data when a specific sensor give an enable signal (i.e. the proximity sensor detect something in range). Once enabled, the gateway gathers all the data at their default data rate. This function is important for Cloud training. By knowing what is the nominal values range for the system, the server is capable of determine whether the values are really out-of-control. The parameters acquired in this mode are shown in Table 1. The ED then remains in *idle state*, for power minimization, until a Smart Sensors sends an Event-signal. When such trigger signal is produced, the ED activates (Event-driven activation) and gathers information. In the demo implemented the SensorTile sends an activity signal. The detected Event from sensor force the ED to read the signals from all the other sensors. In case that, due to the default data-rate the signal of some sensors is not immediately available, the ED sends a read-command to the sensor, in order to have a measure at the time of the event (for instance this is the case of the Heart rate monitor whose default data-rate is 1/20minutes). Once the full IoT sensors data are acquired, the ED evaluates the situation correlating different sensor data, based on local settings (a reduced set) and with the limited local calculation resources (of course the local algorithms have to be defined case by case and are not the target of this proposal). In this design case a function locally evaluates the HR and BP of the subject to determine if they are in the range of value set by the Cloud. The ED then calls the actuator (in this demo it simply gives a pre-warning video output). After this, the ED sends all the data and the pre-warning signal to the server. The Cloud-server reads the pre-warning signal and elaborates a response based on the large Cloud data and large computational power to decide if the local event response is correct. The demo evaluates the temperature and humidity gradient compered to his data-base in order evaluate if the situation is dangerous (i.e. fire hazard) or not. The Cloud-server can then either change and transmit local settings to each Smart-Sensor (so that a better local response can be performed next time) and/or call another actuator.

A. Results

The implemented demo proves innovation as follows:

- *Fast pre-warning response:* as main innovation, the system produces a pre-warning signal interrogating every sensor available after the *Event*. Thanks to the Event response phase, the system reacts and quickly gathers the needed information, unachievable in CLC systems.
- *Local settings:* this system overcomes one of the greatest FC and CoC problem. The Cloud, through specific algorithms, huge computing power, and access the large data bank, can easily do a fine-tuning change of settings for the local system.
- *Power consumption:* the system has proven to implement specific power reduction techniques. For example, one sensor can be called upon just during the event analysis.
- *Data transfer:* the demo also implements a more efficient way to transfer the data. Since a big part of the processing is done locally, only a small part of the raw data is transferred to the Cloud. This is beneficial for situations where internet connectivity is critical.
- *Computational Load:* it was possible to efficiently distribute the computation load and decision-making across the hardware of the system (CoC paradigm). This will prove crucial in heavy computational process, giving the opportunity the manage calculations in a much efficient way.
- *Cloud-independent operation*: the system can operate (based on local settings) also without any Cloud connection.

IV. Conclusions

In this paper, some key limitations in modern CLC IoT systems are presented. Some of these issues can be overcome by means of distributed tasks. In alternative, a CoC based implemented system can be considered as here reported. The implementation results on the network operation show that latency, sensors variety and power consumption limitations of CLC can be overcome by this innovative *Event-driven* IoT model.

V. Acknowledgments

This research is funded by the Lombardia Regional Project “The Home of Internet of Things (Home IoT)” (ID139625)

References

[1] Michèle Royer, PhD “The Internet of Things (IoT) A trends white paper” - August 2013

[2] J. M. Williams et al., "Weaving the Wireless Web: Toward a Low-Power, Dense Wireless Sensor Network for the Industrial IoT," in IEEE Microwave Magazine, vol. 18, no. 7, pp. 40-63, Nov.-Dec. 2017.

[3]S. Karthikeyan and P. T. V. Bhuvaneswari, "IoT based real-time residential energy meter monitoring system," 2017 Trends in Industrial Measurement and Automation (TIMA), Chennai, India, 2017, pp. 1-5.

[4] H. Al-Hamadi and I. R. Chen, "Trust-Based Decision Making for Health IoT Systems," in IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1408-1419, Oct. 2017.

[5] H. Desai, D. Guruvayurappan, M. Merchant, S. Somaiya and H. Mundra, "IoT based grocery monitoring system," 2017 Fourteenth International Conference on Wireless and Optical Communications Networks (WOCN), Mumbai, India, 2017, pp. 1-4.

[6] A. Ciuffoletti, "OCCI-IoT: An API to Deploy and Operate an IoT Infrastructure," in IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1341-1348, Oct. 2017.

[7] P. Mell et al., “The NIST denition of cloud computing," Nat. Inst. Standards Technol., U.S. Dept. Commerce, Gaithersburg, MD, USA, Tech. Rep. 800-145, 2011.

[8] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing," 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Athens, 2014, pp. 325-329.

[9] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of Things," in Proc. 1st Ed. MCC Workshop Mobile Cloud Comput., 2012, pp. 1316.

[10] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou and Y. Zhang, "Multi-tier Fog Computing with Large-scale IoT Data Analytics for Smart Cities," in IEEE Internet of Things Journal, vol. PP, no. 99, pp. 1-1.

[11] Cisco. (Apr. 2015). Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are. Accessed on May 14, 2017. [Online]. Available:http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[12] M. Samaniego and R. Deters, "Internet of Smart Things - IoST: Using Blockchain and CLIPS to Make Things Autonomous," 2017 IEEE International Conference on Cognitive Computing (ICCC), Honolulu, HI, USA, 2017, pp. 9-16.

[13] Y. Sahni, J. Cao, S. Zhang and L. Yang, "Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things," in *IEEE Access*, vol. 5, pp. 16441-16458, 2017.

[14] F. Montori, R. Contigiani and L. Bedogni, "Is WiFi suitable for energy efficient IoT deployments? A performance study," 2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI), Modena, Italy, 2017, pp. 1-5.

[15] http://www.st.com/en/evaluation-tools/steval-stlcs01v1.html