Towards a Resilient Control Architecture for Combined Fog-to-Cloud Systems

Xavi Masip-Bruin, Sergi Sánchez-López
Alejandro Jurnet, Eva Marín-Tordera
Universitat Politècnica de Catalunya, CRAAX-UPC
Vilanova i la Geltrú, Spain

Admela Jukan
Technische Universität Braunschweig
Braunschweig, Germany

Guang-Jie Ren
IBM, Almaden Research Center
San José, USA

***Abstract*—The capacity to efficiently manage the whole set of resources from the edge up to the cloud paves the way to a new landscape of innovative opportunities for all involved actors, be it on the research or industrial sides. Fog-to-Cloud (F2C) has been recently proposed as a management solution particularly tailored to manage the stack of resources from the edge up to the cloud in a coordinated way. However, beyond the benefits brought by considering all the spectrum of resources to run a service, resilience, as a concept must be reflected in the F2C design. In this paper, we address a particular scenario where a specific node failure in the F2C architecture will substantially impact on the whole system performance, and analyse three tentative strategies to efficiently manage such scenario.**

Keywords—Fog computing, resilience, Fog-to-cloud systems.

I. INTRODUCTION AND MOTIVATION

Nowadays, there is an unstoppable trend towards deploying smart systems (e.g. smart cities, smart cars, smart home, smart manufacturing or smart transportation), supporting and enabling the development of innovative added value services. Aligned to the smartness concept, it has been widely accepted that cloud and fog computing play a vital role to facilitate the deployment of highly demanding services (usually linked to smart processing), by providing resources where all processing and storage needs may be met. Reducing either delay or the need to transfer large amounts of data far from the edge device up to the cloud, seems to be the main target, and actually the main driver for fog computing [1] to come up. some references in the literature discuss on the similarities and differences between edge and fog computing [2].

Assuming fog is not competing with cloud, rather collaborating each other to bring some functionalities cloud cannot properly handle (for example low latency), an optimal service execution would require some strategy intended to optimally map services into the set of available resources. This challenge has been already identified, collecting several preliminary efforts, such as the OpenFog Reference Architecture (OFRA) [3], the efforts done in ETSI Multi-access Edge Computing (MEC) [4], particularly focused on the mobility issues inherent to edge devices, and the European H2020 mF2C project [5], intended to design and implement the hierarchical Fog-to-Cloud (F2C) management architecture previously proposed in [6] and evaluated in [7]. Recognized the novelty of these proposals many research challenges are still open, driving notorious efforts on both of them. Indeed, a fundamental issue to assure a successful deployment of any resource management solution, falls into the resilience area.

This paper addresses resilience in an F2C scenario by analyzing: i) the effects a failure in a key control element (leader) in the F2C architecture may have in the overall performance; ii) the causes motivating a failure on such a key component, and, iii) different novel strategies to efficiently recover control information in such a failure scenario.

This paper reads as follows. Section 2 introduces the proposed hierarchical architecture Then, Section 3 proposes three tentative strategies to improve the database hand over process. A preliminary evaluation of the benefits brought by the proposed strategies is introduced in Section 4, right before concluding the paper in Section 5.

II. RESILIENCE IN COMBINED F2C SYSTEMS: LOSING CONTROL

This section starts by revisiting the control architecture proposed in F2C systems. Then, a control failure scenario is described, emphasizing potential causes motivating the failure as well as the effects a failure will bring to the whole system.

A. Control in F2C Systems

F2C envisions a hierarchical architecture putting together the different layers the whole stack of resources is mapped into (see Fig. 1), where each individual layer encompasses a certain set of resources according to a policy (P1) in place (defined in Section III). In a hierarchical approach, nodes or elements in the topology are clustered into different groups led by a single node logically mapped into a higher level in the hierarchy. This node, refereed in F2C to as the leader (L) (see Fig. 1b), is responsible for collecting information about the nodes it has connected and forwarding the collected information upwards in the hierarchy. The processes of nodes clustering, information collection and forwarding are repeated on each layer in the hierarchy. It is with no doubt that the whole process must be handled by a set of policies, included in a management

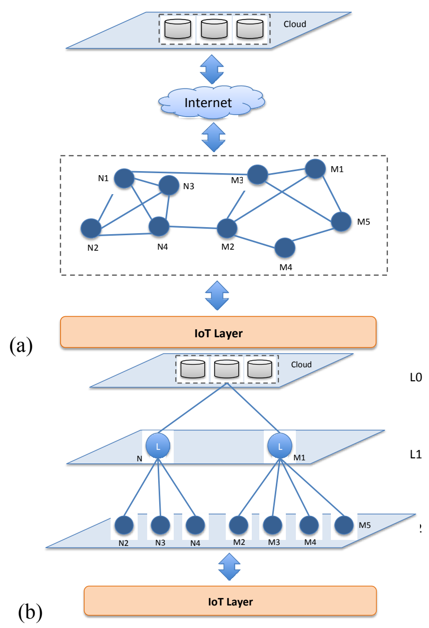


Fig. 1. The resources continuum: a) physical topology; b) hierarchical topology

strategy, ending up in a correct mapping of the physical resources topology into a hierarchical (logical) topology, ready to be properly managed.

For the sake of understanding, let's suppose the illustrative example drawn in Fig. 1. The example depicts both the physical resources topology including a possible set of resources (Fig. 1a), and the hierarchical topology it could be mapped into, once the hierarchical approach is applied (Fig. 1b). Fig. 1a shows a cloud connected to different devices at the edge (IoT layer) though several intermediate nodes, shown as [N1..N4] and [M1..M5]. These intermediate nodes can be either fixed (traffic light, street cabinet, etc.) or mobile (car, mobile phone, etc.). The physical topology in Fig. 1a becomes the hierarchical topology shown in Fig. 1b, by applying several policies, being the most relevant ones the policies to be used to define the layers (P1), to cluster the nodes (P2) and to decide what the leader is for each individual group of nodes (P3). A preliminary approach for these policies is included in Section 3. In the proposed example, policy P2 ends up selecting two groups (consisting in the N and M nodes each), while policy P3 selects one leader for each one of them (N1 and M1). As a consequence, a hierarchical structure is built, showing cloud at the higher layer (L0), the two nodes defined as leader in Layer 1 (L1) and the remaining nodes (refereed to as children) in Layer 2 (L2). Indeed, this is a simple scenario intended to illustrate how the hierarchical topology is built.

The information collected by the leader from its children (may include any data required to suitably manage the whole system, such as available resources, applied policies, users profiling, etc.), is forwarded upwards to the higher node in the hierarchy, usually after an aggregation process intended to reduce the volume of data to be sent. Beyond the potential loss in accuracy, aspect not addressed in this paper, it is worth highlighting the fact that all collected information must be stored in a database located at each leader that must be perfectly synchronized. Thus, the local database located at N1 will include information about nodes N2, N3 and N4, and the same for M1 with M2, M3, M4 and M5. That said, it must be highlighted that aspects related to the databases operation are out of the scope of this paper.

B. *Losing the Control: Causes and Effects*

The policy to be used to select the leaders (P3) must consider robustness as a crucial parameter in the leader selection process. As widely used in many different domains, a proactive solution to mitigate this problem, would require the selection of backup nodes to be used to protect the whole control system from leader failures. There are many approaches in the literature discussing about resilience and specifically addressing the selection of backup instances to properly protect the "primary" element. In this paper, we will consider the 1:1 approach applied to the leader, hence computing a backup for each leader.

However, recognized the fact that F2C deals with highly dynamic and thus volatile devices, the chances for a leader to fail are not negligible. Indeed, a failure here does not only refer to a node misbehavior but rather includes aspects related to non-availability –switch on-off battery policy, node on the move, users policy, etc.–, that are envisioned to be more usual in the devices to be managed in a F2C context. Thus, it is with no doubt that a proper leader selection policy is needed but also a policy to select the backup (P4).

The backup selection process must deal with two aspects. The first is the selection of the physical node matching the requirements to become a leader according to the policy in place. The second refers to the hand over process. In fact, the main issue in the hand over process right after a leader fails, focuses on how and when the information stored in the leader database is moved to the backup, so the latter is endowed from the very beginning with a consistent and accurate view of the physical topology. Also, the faster the information in the backup database is stabilized the lower the effects of a leader failure will be.

III. Handling the Leader/Backup Hand Over

This section introduces both the main assumptions in the overall leader/backup management, and the database management process, proposing three different strategies to guarantee an optimal react to a leader failure.

A. *Global Policies and Assumptions*

As said in Section 2, F2C has been proposed as a control architecture to manage the whole stack of resources in a coordinated way. Although policies definition is an ongoing work, the following policies are identified as mandatory to build the whole management system:

- P1: Layers definition: considers information from the resources categorization and any specific rule from the infrastructure or service providers.
- P2: Devices clustering: different fogs are composed by clustering devices according to an existing policy.

– P3: Leader selection: responsible for choosing the node best suited to act as leader. Parameters to be considered may rely on node stability, power capacity, dynamics, etc.
– P4: Backup selection: considers the expected leader functionalities to guarantee an efficient hand over when needed.

Assuming the set of policies above, some assumptions are also considered to build our scenario.

– The control system is protected by a 1:1 strategy, where each leader has its own backup.
– P3 and P4 are mandatory. In other words the system always finds a leader and a backup for each leader. Consequently, a fog group is only created whether the pair leader-backup exists.
– Leader and backup must not have the same requirements. Indeed, the degree of matching will depend on the real availability and on the expected services performance.
– Different policies can be considered to handle backup availability, particularly devoted to handle the scenario when there are no suitable nodes to serve as backup, or when the selected backup is not available when needed.
– The leader periodically broadcasts sort of "welcome" messages, to both discover new elements and maintain those already linked to it. When the leader goes down, the backup starts broadcasting these messages, thus announcing remaining nodes who the new leader is [9].
– The backup sends sort of "alive" message to the leader to make the former know when the leader is down.
– Periodicity for "welcome" messages is largely lower than the one of "alive" messages, to facilitate the detection of a leader failure.

B. *Strategies for Database Hand Over*

Next, we propose three different strategies addressing the database hand over processing from the leader to the backup. The main objective is to guarantee that once the leader is down, the backup becoming the new leader has the required information to start handling the remaining children with minimum negative effects.

The three proposed strategies depend on how and when the new leader is granted to have as accurate information as possible, so as it could start efficiently managing the remaining nodes in its group. The proposed strategies are Zero-Knowledge (ZK), Keep Updating (KU) and High-Layer download (HLD). The ZK and the HLD strategies both consider that no information is stored in the backup. The KU instead, does it, thus handling a copy of the leader database.

The first action, common to all proposed strategies focuses on the backup selection. To that end we start assuming a policy in place has already appointed a node to serve as backup. Then, the next step is to detect, when the leader is down. Although many different approaches may be designed for that objective, we consider the backup to periodically send a sort of "alive" messages to its leader in order to detect whether the leader is still alive. So far, the process is the same for the three different strategies but it is not from now on.

In the ZK strategy, the backup does not have any database with the leader information, thus when it detects the leader is down, it becomes the leader and forwards a message to the remaining children in the group asking them for state information. As the information from the nodes is received, the new leader will create and update its database. In this scenario some time is required right after the leader is down to set the database in the backup. On the other hand, no messages are needed to keep a database at the backup updated. Finally, a new backup must be selected and once this step is done, the backup starts sending Alive messages to the leader.

In the KU strategy, the backup already has a copy of the leader database that is supposed to be periodically updated to guarantee its accuracy, according to a certain policy to be defined. When the leader fails, the backup may immediately start serving as leader since it already has the required information to run as leader. Then, the backup communicates to all remaining nodes that it is the new leader and keeps using the same updating mechanism deployed by the former leader. In this scenario the time to serve as leader is minimized, but a database synchronization scheme must be deployed.

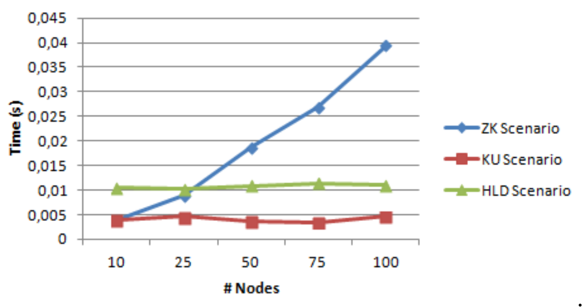
The HLD strategy relies on a high layer node to be the repository for a copy of the leader database. When the leader is down, the backup retrieves the information from the higher layer node (copies the database) and warns all remaining nodes about the new leader. Again, some time is required for the backup leader to serve as leader since the required information is not available from the early beginning. In HLD some effort is also required to keep updates copies of the database information in the higher layer node.

IV. Preliminary Results

In this section we carry out a preliminary analysis of the performance for the proposed database synchronization strategies in terms of time and overhead, considering:

– P1: Three layers are considered aligned to the preliminary studies already presented in[8].
– P2: Fog is composed of a single cluster of devices grouped according to real coverage.
– P3: The leader is selected manually only when setting the system for the first time.
– P4: Backup is selected to be a static element in the topology, and the node selected as backup is always available when the leader goes down.

The topology used for the evaluation consists in 100 nodes, as it is supported by the physical testbed, out of which, to show real devices, 4 nodes are implemented with RaspBerry Pi and 96 nodes are emulated as virtual machines. Nodes connectivity is provided by a private WiFi network with no added traffic. The 96 virtual machines are set using VirtualBox 5.1.28 running on a physical 64-bits computer built on 12 Intel core ES-2620 v2 at 2.1GHz. Each virtual node runs a Debian operating system (32-bit), considering 2 CPU, 8GB HDD and 4GB RAM. The real 4 nodes are deployed using Raspberry Pi 3 model B, all with a Broadcom Chipset Processor BCM2387, 1.2 GHz quad-core ARM Cortex-A53 and 1GB RAM.



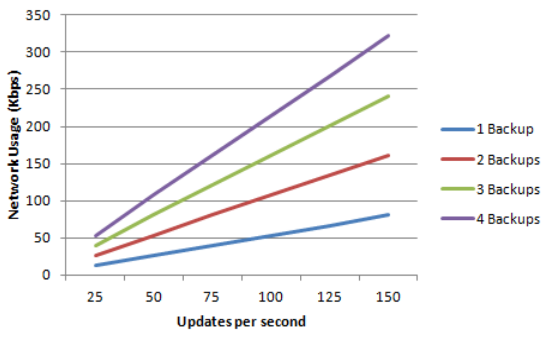


Fig. 2 Results: a) Backup time to update the database; b) Cost in messages overhead for database updating

Fig. 2a shows the time required to update the database at the backup for each one of the proposed strategies. The blue curve represents the time required by the ZK strategy to enable the backup leader to create the database, ask the remaining children for their information and set it all in the database. As expected, this strategy is the most time-consuming, since there is no database created in the backup node. The KU strategy is the fastest in setting the database at the backup. This effect was also expected, since the database is already set and only an updating process is required. Finally, working in between, the HLD strategy shows the time the backup leader consumes to retrieve the database from a high layer node. Thus, from a time perspective, the most efficient strategy is the one based on periodically and continuously updating a database in the backup leader with information from the leader. However, this strategy would require a significant cost in terms of the amount of messages to be disseminated throughout the network to synchronize both databases. This effect is shown in Fig. 2b.

Fig. 2b is intended to show two different characteristics. First, the blue line shows the overhead in terms of updating messages forwarded to the network for the KU and HLD scenarios (the ZK strategy would not require such update messages). Assuming 67 bytes (consisting in the UDP header and the required information we use to update the database) to be sent on each updating message to the backup, the figure shows the network occupancy as per the number of updates. We can observe in Figure 6a that the KU scenario is the one that has a better behavior with respect to the update time of the topological database. However, if we observe Fig. 2b we see that this scenario produces an effect in the use of the network (blue line), that is, an increase in overhead messages with respect to the ZK scenario (does not need any updating). Therefore, it will be necessary to establish a compromise between the update time of the topological database and the load effects of the network in order to choose the best scenario.

We may conclude that for a low amount of devices ZK is the one performing better since turns into a reduced updating time while driving no overhead. However, as the number of devices goes up, the strategy performing the best will be either KU or HLD accruing to the services needs and the user expected quality.

Second, Fig. 2b is also intended to show the effects of selecting more than one backup leader –in the same group. Indeed, the figure draws the trend regarding the effects on the network usage when considering 2, 3 and 4 backup leaders (red, green and purples lines respectively), for the KU strategy only, since the number of backups does not affect the HLD strategy where update messages are only sent to cloud (or a node in a higher layer). In a real scenario the update policy to be selected must guarantee accuracy as long as the system overhead becomes reasonable.

V. Conclusions

The paper proposes three strategies, aimed at guaranteeing an accurate database hand over when reacting to a node failure. The preliminary set of results included in the paper, concludes that the Keep Updating (KU) strategy is the one requiring lower synchronization time but higher overhead. Results also analyze the effects of considering several backup options for each strategy, highlighting the impact on the network utilization. Finally, future efforts will basically go into two key aspects, policies definition and resources categorization.

Acknowledgment

This work was partially supported by the H2020 mF2C project 730929 and for the UPC authors also by the Spanish Ministry of Economy and Competitiveness and by the European Regional Development Fund, under contract RTI2018-094532-B-I00

References

[1] F. Bonomi, et al., Fog Computing: A Platform for Internet of Things and Analytics, N.Bessis and C.Dobre (eds.), Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence 546, DOI: 10.1007/978-3-319-05029-4_7, Springer 2014

[2] E.Marín-Tordera, et al., Do we all really know what a Fog Node is? Current trends towards an open definition, Computer Communications, Vol. 109, pp. 117-130, September 2017

[3] Open Fog Consortium Working Group , "OpenFog Reference Architecture for Fog Computing" White paper, Feb 2017

[4] ETSI, Multi-access Edge Computing (MEC) http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing, [Accessed: February 2018].

[5] mF2C project, http://www.mf2c-project.eu, [Accessed: February 2018]

[6] X.Masip-Bruin, et al., Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud (F2C) computing systems, Wireless Communication Magazine, Vol. 23, Issue 5, October 2016.

[7] W.Ramirez, et al, Evaluating the Benefits of Combined and Continuous Fog-to-Cloud Architectures, Computer Communications, Vol.113, pp. 43-52, November 2017

[8] E.Marin, et al., “A hierarchical routing approach for optical transport networks”, Computer Networks, Vol.50, nº:2, pp. 257-267, Feb. 2006

[9] Z.Rejiba, X.Masip-Bruin, E.Marín-Tordera, G.J.Ren, “F2C-Aware: Enabling discovery in Wi-Fi-powered Fog-to-Cloud (F2C) systems", IEEE Mobile Cloud, Bamberg, Germany, March 2018