A Decomposition-based Architecture for Distributed Cyber-Foraging of Multiple Edge Functions

Flavio Esposito
Saint Louis University
flavio.esposito@slu.edu

Federica Paganelli
CNIT
federica.paganelli@cnit.it

Romano Fantacci
University of Florence
romano.fantacci@unifi.it

Abstract— **Edge computing is an emerging paradigm whose goal is to boost with proximity cloud resources the computational capability of otherwise weak devices. This is also useful to reduce user perceived access latency to backend service. A central mechanism in edge computing is cyber-foraging, *i.e.*, the search and delegation to capable edge cloud processes of tasks too complex, time consuming or resource intensive to be running on user devices or low-latency demanding to be running remotely, as a form of edge function. An edge function is any network or device-specific process that may be run on an edge process instead. Despite the recent interest for this technology from industry and academia, cyber-foraging techniques and protocols have yet to be standardized.**

In this paper, we leverage decomposition theory to propose an architecture whose aim is to provide insights in the design and implementation of protocols for cyber-foraging of multiple edge functions. In contrast with several existing solutions, we also argue that the (distributed) cyber-foraging orchestration should be policy-based and not ad-hoc solution, as opposed to either a pure edge cloud burden or a device decision. To this end, via simulations and leveraging decomposition theory, we show how our approach can be used by edge computing providers and application programmers to compare and evaluate different alternative cyber-foraging solutions. Our decomposition-based approach has general applicability to other network utility maximization problems, even outside the edge computing domain.

I. Introduction

The Internet of Things (IoT) paradigm supports interaction with sensing and actuating devices distributed in the physical environment, thus fostering the development of novel applications and services in several domains *e.g.*, healthcare, disaster response [3], home automation, or automotive, to name a few. Typically, IoT deployments leverage cloud computing to complement constrained local resources with remote processing and storage capabilities (Data Centers). To support the increasing demand for low-latency, responsive and resource intensive applications, a fork of the cloud computing paradigm (Mobile Edge Computing) has seen the evolution of heterogeneous, federated clouds distributed at the edge of the network. Edge computing [14] is enabled by the ability to run powerful servers at the edge of the network *e.g.*, at a base station site, thus allowing applications hosted on those edge servers to deliver low-latency and responsive *edge functions* *i.e.* software capabilities run at the edge of the network to leverage the advantages of user proximity and local network information.

This work has been partially supported by the National Science Foundation award CNS-1647084.

In such scenario, distributed applications that require low latency communications and context-based adaptation and exploit such distributed resource infrastructure can be conceived, also leveraging proximity to end users, data sources (e.g., sensors) and actuators. For instance, an application could be deployed as a composition of *functions* that process data and video streams delivered by sensors, *e.g.* a microscope camera being controlled remotely in a tactile Internet, a phone or UAV camera hovering a disaster scenario looking for survivals [3], or a vehicle inferring the status of the road sending alerts to other subscriber vehicles. Moreover, leveraging the current evolution trend in networking towards network programmability (Software-Defined Networking - SDN) and virtualization of network functions (Network Function Virtualization - NFV [12], [6]), specific traffic handling requirements and Service Level Agreements could be flexibly handled through appropriate network orchestration mechanisms [18] (a video optimizer could be deployed to handle the video streams originated by webcams). In a challenged edge computing scenario, *e.g.*, for natural or man-made disaster response, or for Internet of Medical Things (IoMT) applications, multiple processes need to establish and maintain a set of virtual flows to guarantee a set of Service Level Objectives, *i.e.*, some acceptable levels of network performance, to accomplish a phase or, more generally, to provide a service. Several edge cloud infrastructures arose in the research community, see *e.g.* [2], [5], [15], [20], [21], as well as from industry initiatives, such as ETSI Mobile Edge Computing [10].

Our contribution. In contrast with the vast majority of these proposals, that either choose back-end driven onloading [7], [16] or mobile-driven offloading [2], [5], [15], [20], we argue that leaving the flexibility of a choice to application programmers and edge computing providers is a wiser alternative, since it allows decisions to be made considering several aspects, such as the type of process (*e.g.*, I/O or computation intensive), available orchestration architectural options, as well as application and edge infrastructure providers' goals. In some cases, client-driven offloading algorithms are the only available option [10], while in other cases they face many challenges, due to the diversity of devices and to their scarce computational resources available: (i) different operating systems need to cooperate across a (ii) plethora of numerous apps, trying to run (iii) accurate code profiling, and (iv) gauging optimal offload conditions. Moreover, several related solutions focus on the orchestration of a single edge function [2], [5], [16].

To capture the benefits of both alternative cyber-foraging orchestration architectures, and the need of a more holistic cyber-foraging orchestration, in this paper we propose a policy-based architecture that leverages decomposition theory to model the (holistic) cyber-foraging of multiple edge functions. We analyze via simulations alternative decomposition policies to evaluate competing (distributed) orchestration of both user and edge cloud resources.The goals of our architecture were (i) to identify the minimum set of mechanisms in the cyber-foraging problem, (ii) to propose an edge computing protocol design tool that could be used to evaluate and compare possible cyber-foraging architectures.

Paper organization. The remainder of the paper is organized as follows: in Section II we discuss related work. In Section III we introduce the problem, in Section IV we discuss how primal and dual decomposition techniques can be applied to solve it. In Section V we propose two iterative solutions based on primal and dual decomposition. In Section VI we analyze the tradeoffs between the two proposed primal and dual decompositions solutions in terms of optimality, convergence speed and signaling overhead and we conclude our work in Section VII.

II. Related Work

Offloading Algorithms. Although the problem of function placement over a distributed resource infrastructure has been floated before, such as across distributed data centers [11] and in network infrastructures [13], only recently some authors have begun tackling such problem in an Edge/IoT environment. Here we only show significant related work to highlight our contributions. We classify these solutions in two categories: *i)* solutions that focus on the problem of offloading from mobile devices to the edge cloud, and *ii)* solutions that focus on workload distribution among edge nodes. In the first category, the burden of offloading decision is usually placed on the end-user device. MAUI [5] and CloneCloud [2] provide different optimization strategies for the migrating part of the workload from a mobile device to a server on the edge cloud. ThinkAir [20] is a framework that allow to offload mobile computation on multiple virtual machines exploiting parallelization. In the second category, the offloading decision is typically assigned to a cloud manager in either a centralized or distributed fashion. Our work differs from these, since we do not argue for one solution, but rather we propose a decomposition theory-based architecture that can evaluate both approaches by merely instantiating a few decomposition policies.

Decomposition Theory. Decomposition theory has been used as a tool for architecting other network utility maximization problems before [17], [8], [9]. In [9] for example, authors use decomposition theory to solve the virtual network embedding, a constrained graph matching problem modeled as network utility maximization. Other solutions used decomposition theory for network utility maximization problems, although most of their focus is on optimization for scarse wireless resources [17] or scheduling for grid computing [4].

Our network utility maximization problem is different as it uses decomposition theory to solve the cyber-foraging of multiple edge functions, and takes a step forward from the system architecture point of view by introducing a unifying architecture, whose policies lead to several distributed cyber-foraging solutions.

III. Problem Statement

Let us consider a network with n physical or virtual edge computing nodes, with index set $\mathcal{I} = \{1, \ldots, n\}$, which are required by the edge computing infrastructure to run a set of (offloaded or onloaded) tasks or edge functions. These edge functions have been sometimes considered in a (chained) order, other times as standalone jobs to run at the edge rather than on the IoT device. Edge functions may require implementing network mechanisms or preprocessing of application data. Examples of (virtual) network edge functions, may be a stateful firewall, a deep-packet inspection process, or a load balancer, while examples of applications functions may be histological image preprocessing, data partitions or aggregations for very large database queries, real-time (vehicle's plate) image recognition, video encoding and compression, or even a public key encryption.

Edge function mapping requests may arrive at any time in an online fashion. We assume that each edge computing node i can be assigned at most a bundle of edge functions $\mathcal{B}_i(t)$, as a subset of the resources available at time t. An edge function bundle $\mathcal{B}_i(t)$ to run on the edge process $i \in \mathcal{I}$ is a list of (virtual) functions that must be executed by the edge computing infrastructure, in a strict order (if it's a service function chain) or in parallel (if it's a single expensive process that we plan to onload using cyber-foraging. We define the time-varying set of overall edge tasks at t as $\mathcal{J}(t) = \cup_{r=1}^{R} \mathcal{S}_r(t)$, where S_r is the set of edge functions within request $r \in \{1, \ldots, R\}$ to the edge computing infrastructure. Due to the online nature of the edge function request arrival, $\mathcal{J}_i(t)$ is a stochastic function. When sampled, $\mathcal{J}_i(t)$ returns a vector (or list) of edge function identifiers to run. We assume that edge processes consider only functions in $\mathcal{J}(t)$, and they are unaware of subsequent requests. Due to its connectivity, end-user processes may be unable to offload computations to some edge node. Similarly, every edge node has a limited capacity and hence is only capable to onload a function from a limited number of processes *e.g.*, geographically close, or following any other distance function.

Given a planning time horizon H, and a non-negative cost function $c_{ij}(t) : \mathcal{I} \times \mathcal{J}(t) \to \mathbb{R}$ indicating the system cost when the edge node i is assigned to the computational edge function j, the *online multi-edge function cyber-foraging problem* can be stated as the following stochastic binary (mixed-integer linear) program:

$$\min_x \sum_{t=0}^{H} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}(t)} c_{ij}(t)\, x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j \in \mathcal{J}(t)} x_{ij} \le |\mathcal{B}_i(t)| \quad \forall i \in \mathcal{I} \tag{2}$$

$$\sum_{i\in\mathcal{I}} x_{ij} \le 1 \quad \forall j \in \mathcal{J}(t) \tag{3}$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in \mathcal{I}\times\mathcal{J}(t) \tag{4}$$

Considered the scenario described above, in this paper we solve the following problem:

***Problem** 3.1: Given the network of n edge servers and the time varying set of edge functions $\mathcal{J}(t)$, solve problem (1)-(4) in a decentralized fashion.*

Reducing it from the set packing problem, it is easy to show that even the deterministic version of Problem (3.1) is NP-Hard. In the next section we show how, instead of finding heuristics to solve our problem, we rely on decomposition theory and on the the interior point method. In the next section we describe how we can use decomposition theory to provide a policy-based architecture to solve Problem (3.1).

IV. Decomposition Architecture

A single process may solve Problem (3.1) logically centralized; in some cases, a logically centralized solution is preferable. In other cases though, a distributed solution is more appropriate, $e.g.$, to avoid a single point of failure, or to allow federated edge cloud providers to collaborate (or compete) in the resource allocation process. In this paper, we are interested in allowing edge cloud processes to solve the problem either in a centralized or in a distributed way, by merely choosing a (decomposition) policy. By policies we mean the variant aspects of any of the decomposition mechanisms (invariances), that is, dual decomposition, primal decomposition, Bender decomposition, or a combination of them, iteratively applied on different versions of the subproblems. Each alternative decomposition leads to a different distributed algorithm, with potentially different desirable properties. The choice of the adequate decomposition method and distributed algorithm for a particular sub-problem depends on the application as well as on the edge infrastructure provider goals. The idea of decomposing Problem (3.1) is to convert it into equivalent formulations, where a master problem interacts with a set of subproblems. Decomposition techniques can be classified into *primal* and *dual* [1]. Primal decompositions are based on decomposing the original primal problem (3.1), while dual decomposition methods are based on decomposing its dual. In a primal decomposition, the master problem allocates the existing resources by directly assigning to each subproblem the amount of resources that it can use. Dual decomposition methods instead correspond to a resource allocation via pricing, $i.e.$, the master problem sets the resource price for all the subproblems, that independently decide if they should use their resource to host a task or not, based on such prices.

Primal decompositions are applicable to problem (3.1) by an iterative *partitioning* of the decision variables into multiple subsets. Each partition set is optimized separately, while the remaining variables are fixed. For example, we could apply a primal decomposition policy by first solving Problem (3.1) with respect to the set of edge functions that deal with load balancing (without loss of generality, we can

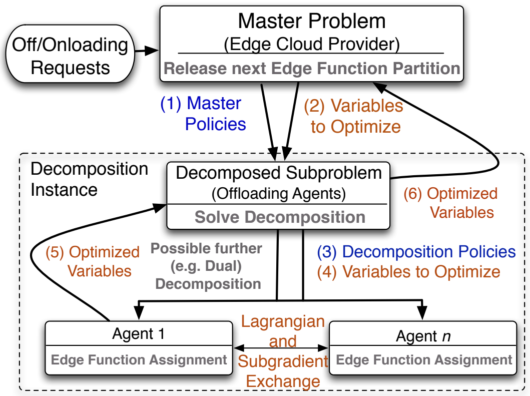

Fig. 1. Architecture Overview: different cyber-foraging solutions can be modeled via primal and dual decompositions. A logically centralized controller may instantiate a problem formulation according to its policies by (1) picking an objective function c (2). Processing agents solve the decomposed subproblems, possibly further decomposing them (3-4). Finally, the optimal variables are returned to the cyber-foraging orchestrator (5-6), that eventually releases the next problem partition variable set.

assume that those are the first k decision variables), and then optimize with respect to the remaining $k-n$ variables, referring to all other virtual edge function services, given the optimal values of the first k variables. Alternatively, a distributed multi-edge function cyber-foraging algorithm could simultaneously optimize all edge functions under the control of a single edge computing process (after electing a leader with a decentralized consensus algorithm such as Raft [19]), and subsequently optimize the other variables under control of other providers. Primal decompositions can also be applied with respect to different horizons. For example, by fixing each time interval, the problem can be solved by optimizing the edge function that require more frequent operation, for example packet compressions or encryptions, and then later, given those optimal values of the vector $\mathbf{x}_{fast}$, minimizing the onloading or offloading costs for functions that run less frequently, such as routing or stateful firewall (run only at the beginning of each connection setup).

Dual decomposition approaches are based on decomposing the Lagrangian function formed by augmenting the master problem with the relaxed constraints. Also in this case, it is possible to obtain different decompositions by relaxing different sets of constraints, hence obtaining different distributed cyber-foraging algorithms. For example, by relaxing constraints (2), we can model solutions that separate the edge function feasibility subproblem from the final assignment subproblem. Regardless of the number of constraints that are relaxed, dual decompositions are different than primal in the amount of required parallel computation (all the subproblems could be solved in parallel), and the amount of message passing between one phase and the other of the iterative solution method. The dual master problem communicates to each subproblem the shadow prices, $i.e.$, the Lagrangian multipliers, then each of the subproblems (sequentially or in parallel) is solved, and the optimal value is returned, together with the subgradients. It is also possible to devise solutions using both primal and dual decompositions.

In general, an edge cloud infrastructure process may instantiate a set of policies at the *master problem*, after receiving an offloading request, dictating the order in which the variables need to be optimized and on which (geolocated) partition of the edge cloud. The subproblems resulting from the decomposition can also instantiate other sets of decomposition policies, to decide which variables are to be optimized next, in which order, or even further decomposing the subproblems, as shown in Figure 1.

V. Decompositions Tradeoff

Every optimization problem can be decomposed to be solved in either a centralized or distributed fashion. In this section we analyze the tradeoffs between primal and dual decompositions, for a sample subproblem, and we propose a subgradient distributed algorithm to solve it. We later use this case study to show the results of a tradeoff analysis between optimality and speed of convergence of the iterative method used by a CPLEX solver. As a use case study, we consider Problem (3.1) formulated in a standard form, spliting the variables in two partitions. The problem can be formulated as follows:

$$\max_{\mathrm{u,v}} \quad c^T u + \tilde{c}^T v$$

$$\text{subject to} \quad Au \le b \tag{5a}$$

$$\tilde{A}v \le \tilde{b} \tag{5b}$$

$$\mathcal{M}u + \tilde{\mathcal{M}}v \le h \tag{5c}$$

where u and v are the sets of decision variables referring to the first and to the second problem partition, respectively; $\mathcal{M}$ and $\tilde{\mathcal{M}}$ are the matrices of capacity values for the tasks in the two partitions, and h is the vector of all robot capacity limits. The constraints (5a) and (5b) capture the separable nature of the problem into the two partitions. Constraint (5c) captures the complicating constraint.

Distributed Edge Function Stochastic Cyber-Foraging by Primal Decomposition. By applying primal decomposition to problem (5), we can separately solve two subproblems, one for each set of edge functions, by introducing an auxiliary variable z, that represents the fraction of physical and virtual resource allocated to each subproblem. The original problem (5) is equivalent to the following master problem:

$$\max_{\mathrm{z}} \ \phi(z) + \tilde{\phi}(z) \tag{6}$$

where:

$$\phi(\mathrm{z}) = \left\{ \sup_u c^T u \mid Au \le b, \mathcal{M}u \le z \right\} \tag{7}$$

and

$$\tilde{\phi}(\mathrm{z}) = \left\{ \sup_v \tilde{c}^T v \mid \tilde{A}v \le \tilde{b}, \tilde{\mathcal{M}}v \le h - z \right\} \tag{8}$$

The primal master problem maximizes the sum of the optimal values of the two subproblems, over the auxiliary variable z. After z is fixed, the subproblems (7) and (8) are solved separately, sequentially or in parallel, depending on the edge function requirement. The master algorithm updates z, and collects the two subgradients, independently computed by the two subproblems. To find the optimal z, we use a subgradient method. In particular, to evaluate a subgradient of $\phi(z)$ and $\tilde{\phi}(z)$, we first find the optimal dual

Algorithm 1 Distributed Stochastic cyber-foraging by Primal Decomp.

```
1: Given $z_t$, solve subproblems to obtain optimal cyber-foraging $\phi$ and
   $\tilde{\phi}$ for each partition, and dual vars $\lambda^\star(z_t)$ and $\tilde{\lambda}^\star(z_t)$
2: Send/Receive $\phi$, $\tilde{\phi}$, $\lambda^\star$ and $\tilde{\lambda}^\star$
3: Master computes subgradient $g(z_t) = -\lambda^\star(z_t) + \tilde{\lambda}^\star(z_t)$
4: Master updates resource vector $z_{t+1} = z_t - \alpha_t g$
```

variables $\lambda^\star$ for the first subproblem subject to the constraint $\mathcal{M}u \le z$. Simultaneously (or sequentially), we find the optimal dual variables $\tilde{\lambda}^\star$ for the second subproblem, subject to the constraint $\tilde{\mathcal{M}}v \le h - z$. The subgradient of the original master problem is therefore $g = -\lambda^\star(z) + \tilde{\lambda}^\star(z)$; that is, $g \in \partial(\phi(z) + \tilde{\phi}(z))$. For the proof, please refer to §5.6 of [1]. The primal decomposition algorithm, combined with the subgradient method for the master problem is repeated, using a diminishing step size, until a stopping criterion is reached (Algorithm 1).

Distributed Edge Function Stochastic Cyber-Foraging by Dual Decomposition. The optimal Lagrangian multiplier associated with the capacity of the agent i, $-\lambda_i^\star$, tells us how much worse the objective of the first subproblem would be, for a small (marginal) decrease in the capacity of edge process i. $\tilde{\lambda}_i^\star$ tells us how much better the objective of the second subproblem would be, for a small (marginal) increase in the hosting capacity of edge process i. Therefore, the primal subgradient $g(z) = -\lambda(z) + \tilde{\lambda}(z)$ tells us how much better the total objective would be if we move some edge functions to be offloaded from one subsystem to the other. At each step of the subgradient method, more resources (*e.g.*, edge server capacity) of each process is allocated to the subproblem with the larger Lagrange multiplier. This is done with an update of the auxiliary variable z. The resource update $z_{t+1} = z_t - \alpha_t g$ can be interpreted as shifts of some of the functionalities to offload to the subsystem that can better use it for the global utility maximization. The analysis of cyber-foraging by dual decomposition is similar to the primal, and we do not show it for lack of space.

VI. Evaluation

The goal of our evaluation is to assess how different decomposition policies impact the design of an edge function offloading (distributed) protocol. In particular, we analyze with a CPLEX solver the tradeoff between optimality, speed of convergence, and signaling overhead of the primal and dual decompositions solved by the iterative methods described in Procedure 1. We were able to reproduce our results with different size, and we only show here representative example of our simulation campaign, when we onload a set of 50 edge function on an edge cloud of 100, 200, and 500 agents. Since it is always feasible to have a solution in which an edge function is assigned to a server that has no more residual capacity, the *Slater condition* [1] is satisfied for our Problem (6). This means that there is no duality gap *i.e.*, the difference between the primal and dual solutions is zero. Nevertheless, for many latency-sensitive IoT applications is not desirable to wait for the optimal offloading assignment when the improvements relative to the previous iterations are small. Hence, using a diminishing

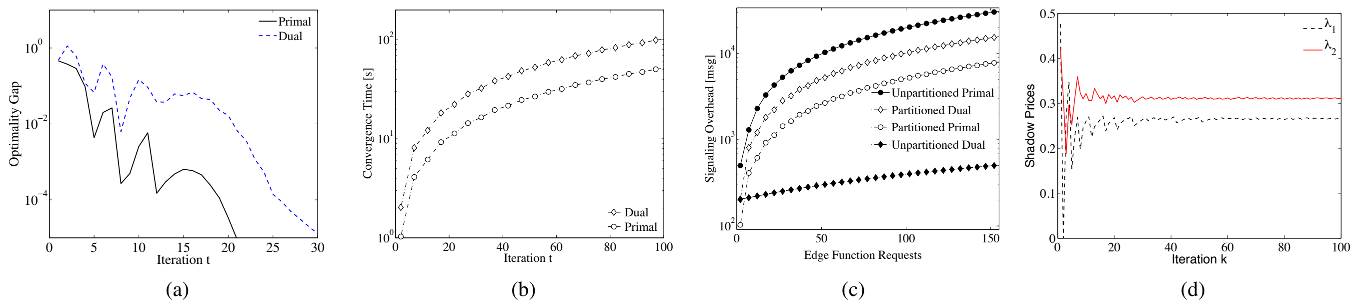

Fig. 2. Using a diminishing step size rule $\alpha_t = 0.5/t$ to complete the first iterations of an edge function cyber-foraging problem solved by primal decomposition leads to a smaller optimality gap (a), at the expense of a larger computational time for each iteration (b), and for a larger signaling overhead among the agents (c). (d) Evolution of the shadow prices λ (d) (and of the resource vector z (e)) during a distributed cyber-foraging subgradient algorithm using dual (primal) decomposition.

step size rule $\alpha_t = 0.5/t$, where t is the iteration step, we stopped our simulations after a fixed number of iteration (Figure 2). We observe that under a primal decomposition policy, the solution reduces its deviation from the optimal solution (known as *optimality gap*) in fewer iterations, albeit at the cost of larger computation time per iteration, and a smaller signaling overhead (Figures 2c).

We also run our simulations with and without partitioning the requested set of edge functions into multiple subsets. Partitioning the set of edge functions means running such iterative method multiple ($N_t/2$) times, but on problems with smaller input size. We applied a partitioning policy of $N_t/2$ partitions, of two edge functions each, where N_t is the total number of requested edge functions to onload on the edge cloud (Figure 2c). Note that even when a set of edge functions is not partitioned, the distributed iterative solution method used for either primal or dual decompositions requires the passing of messages between the master and the dual subproblems.

VII. Conclusions

In this paper we modeled with optimization theory the edge function cyber-foraging problem, a crucial problem that integrates data intensive and latency sensitive IoT applications with edge clouds. We then proposed to solve the utility maximization problem using decomposition theory, and we showed how our approach can provide insights into a systematic design of distributed cyber-foraging solutions. Using our CPLEX-based simulator, we showed how our decomposition approach can be used to analyze key edge function cyber-foraging protocol design tradeoffs. We found how some decomposition policies may lead to a quicker reduction of the optimality gap over the iterations of the distributed solution, at the expense of larger computation time per iteration, as well as a larger signaling overhead.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. http://www.stanford.edu/people/boyd/cvxbook.html, 2004.

[2] Byung-Gon C. et al. Clonecloud: Elastic execution between mobile device and cloud. In *Proc. of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011.

[3] D. Chemodanov, F. Esposito, A. Sukhov, P. Calyam, H. Trinh, and Z. Oraibi. AGRA: Ai-augmented geographic routing approach for iot-based incident-supporting applications. *Future Generation Computer Systems*, pages –, 2017.

[4] L. Chunlin and L. Layuan. Optimization decomposition approach for layered qos scheduling in grid computing. *J. Syst. Archit.*, 53(11):816–832, 2007.

[5] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: Making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 49–62, New York, NY, USA, 2010. ACM.

[6] F. Esposito. Catena: A Distributed Architecture for Robust Service Function Chain Instantiation with Guarantees. In *IEEE 3rd Confeence on Network Softwarization (NetSoft 2017)*, Rome, Italy, July 2017.

[7] F. Esposito, A. Cvetkovski, T. Dargahi, and J. Pan. Complete edge function onloading for effective backend-driven cyber foraging. In *Proceedings of the 13th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, WiMob 2017, Oct 2017.

[8] F. Esposito, I. Matta, and V. Ishakian. Slice Embedding Solutions for Distributed Service Architectures. *ACM Computing Surveys*, Accepted November 2012 (to appear in Volume 46, Issue 1, March 2014).

[9] F. Esposito, I. Matta, and Y. Wang. VINEA: an architecture for virtual network embedding policy programmability. *IEEE Trans. Parallel Distrib. Syst.*, 27(11):3381–3396, 2016.

[10] ETSI. Mobile Edge Computing https://goo.gl/zgnft4, 2017.

[11] L. Gu, D. Zeng, A. Barnawi, S. Guo, and I. Stojmenovic. Optimal task placement with qos constraints in geo-distributed data centers using dvfs. *IEEE Transactions on Computers*, 64(7):2049–2059, July 2015.

[12] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[13] J. G. Herrera and J. F. Botero. Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, Sept 2016.

[14] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young. Mobile edge computing: A key technology towards 5g. *ETSI White Paper*, 11(11):1–16, 2015.

[15] Ketan Bhardwaj et al. AppFlux: Taming App Delivery via Streaming. In *Usenix Conf. on Timely Res. in Oper. Systems (TRIOS)*, 2015.

[16] Ketan Bhardwaj et al. Fast, scalable and secure onloading of edge functions using airbox. In *IEEE/ACM Symposium on Edge Computing*, 2016.

[17] R. C. M. Chiang, S.H. Low and J. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proc. of IEEE*, 95(1):255–312, Jan 2007.

[18] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi. Latency-aware composition of virtual functions in 5g. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–6, April 2015.

[19] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, June 2014.

[20] S. Kosta et al. ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *2012 Proceedings IEEE INFOCOM*, pages 945–953, March 2012.

[21] M. Satyanarayanan, P. Bahl, R. Caeres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.