Distributed Fair Randomized (DFR): a Resource Sharing Protocol for Fog Providers

Roberto Beraldi* and Hussein Alnuweiri†

* DIAG, La Sapienza University of Rome, Italy † Texas A&M University at Qatar, Doha

***Abstract*—Fog computing promises to support many emerging classes of applications that can't be rely on a cloud-only backend. Fog-to-Fog (F2F) cooperation is suggested in the openFog's Fog computing Reference Architecture, now adopted as an IEEE standard, as a way to improve the computation service provided by this computing delivery model.**

In this paper, we propose DFR - Distributed Fair Randomized, a distributed F2F cooperation algorithm that allows for sharing computation resources among fog providers that agree on a (reasonable) measure of fairness. We adopt an analytical approach to study the cooperation problem of providers subject to different load conditions. We initially put the cooperation problem in the light of a simple game-theory framework to capture the selfish behavior of providers without any fairness criteria and its consequence in limiting cooperation. Then, we cast the problem as an optimization problem that incorporates fairness. Preliminary simulations results show how DFR converges to the predicted optimal value.

I. Introduction

Today, almost every software or data application relies on cloud computing. Big players, like Google, Amazon, IBM, Microsoft are then naturally extending their portfolio to offer services embracing new classes of applications based on smart objects and enter in this new promising market. For example, specific solutions are offered by AWS (Amazon Web Service) to implement a smart city and related services or even to support Industry 4.0. The recent release of the Alexa SDK, Google's Smart Home SDK, together with Android Things and the Microsoft's Sphere Operating System, targeting micro controllers are clear signs of a momentum for the IoT ecosystem. It is widely recognized, among industries and academia, that Cloud computing, as is, can't sustain all the forthcoming advanced applications, given the expected proliferation of connected devices, more than 50 billions by 2020, according to many predictions.

Fog Computing (or Edge computing when limited to the telco edges, e.g., Radio Access Network) is the full-blown paradigm that aims to overcome the limitations of this cloud-only, cloud-centric architecture and to open new market opportunities. Fog computing will enable, beside IoT based applications, other verticals, most notably based on 5G, Artificial Intelligence (AI), Virtual Reality and other use cases as described in the 451 openFog's report.

The key idea of fog computing is to create an intermediate layer between cloud and data sources, where some critical part of the new applications can be deployed. As such, it is one of the four layers envisioned in the general distributed computing landscape of the next years[1]. This layer brings calculation, storage, communication, control, and decision making closer to the edge of the network where data is generated and used [2]. By pushing computation and storage towards the network edge fog computing reduces end-to-end communication delay of one/two oder of magnitude, as well as the core network overhead [3], [4].

OpenFog consortium, recently merged with the Industrial Internet Consortium (IIC), describes a Fog Computing Reference Architecture (RA), [5], now adopted as the basis for the IEEE-1934 standard. The European Telecommunications Standards Institute, ETSI, also has released a new specification of its Multi-access Edge Computing (MEC) architecture early this year [6], while NIST describes the fog computing model in its Special Publication 500-325. These initiatives aim at boosting interoperable implementations of fog computing. Under the research point of view, fog computing poses several challenges, e.g., see [7] for a discussion. An open research topic in this field is resource sharing. As the very reason of fog nodes is to provide computation resources locally, horizontal resource sharing, i.e., cooperation among nearby nodes, possibly deployed and managed by different providers, is envisioned in the RA. Also, in the NIST's fog computing model, cooperation among different providers is listed as a characterizing feature of this computation model. Properties like computation elasticity can highly benefit from this kind of cooperation.

In this paper we focus on horizontal resource sharing or Fog-to-Fog (F2F) cooperation. With this term we mean the cooperation among a set of independently managed and nearby fog nodes. This model of resource sharing has its own characteristic. From one hand providers do have interests in sharing their resources as this reduces the risk that a computation cannot be handled in the fog node, for example due to a temporary overload condition. On the other hand, resource sharing via a centralized orchestrator is contrary to the distributed architectural principle of this model, and the selfish behavior has to be taken into account.

We propose DFR, Distributed Fair Randomized resource sharing protocol, a protocol for F2F cooperation. Nodes running DFR agree to borrow each other computation resources for the purpose of absorbing temporary load peaks. The goal of this type of resource sharing is not to sell resources and

978-1-7281-1796-6/19/$31.00 ©2019 IEEE

maximize a revenue, but rather exchange resource usage over time, although this indirectly means an increased utility of providers.

We formalize this characteristic through the notion of *fair* cooperation, the key distinguish feature of this protocol. In simple words, fair cooperation is a resource barter over time among providers. This notion follows a simple reciprocal principle: a provider borrows an average computation capacity to other providers in the group, and gets the same amount of resource back from the providers in the group in the near future. The utility of a computation resource is not bound to any speculation logic and it is constant over time. If all servers have the same computation power, fair cooperation boils down to borrow each other server for the same amount of time. DFR maximizes this time.

The other characteristic of the protocol is that it is based on a decentralized randomization protocol. This means, that when a node is congested and cannot serve a task, the node probes other node in the group at random. This protocol implements a power-of-two choices algorithm that is known to have excellent performance, see [8]. The contribution of the paper is twofold:

- We formalize and study the resource sharing problem among N homogeneous fog providers in two frameworks, game-theory and multi-objective optimization, and provide a closed solution for $N = 2$;
- We propose DFR, a distributed protocol for *fair* cooperation that leverages the finding of the formal study. We provide numerical and empirical simulation results that the algorithm converges to the optimal solution.

The not homogeneous case is left as future work. DRF can be easily extended to cover this case. The remainder of the paper is divided as follows. We present the related work in Section II. Section III discusses F2F resource sharing in general, while Section IV introduces the notion of fairness. In section V we propose DFR. We conclude and discuss limitations and future work in Section VI.

II. Related Work

In the literature, F2F cooperation is recently getting attention of multiple stakeholders as a way to improve cost-performance trade-off. However cooperation among fog nodes were largely limited to cooperation within the same provider/stakeholder [9], [10], [11]. Collaborative offloading schemes of unprocessed workload are proposed to reduce end-to-end latency [10] and the Quality of Experience (QoE) of edge computing registered users [9]. Within the context of the same providers, some have proposed schemes requiring periodic exchange of control messages to a central node, or a broadcast to all nodes to make efficient compute and scheduling decisions [11].

In [12] we studied cooperation among different providers and proposed a virtual coin based protocol as an incentive mechanism to deal with selfish behaviors. The protocol requires resource price adaptation and a central authority. The work of this paper is connected to our previous work [13] where the idea of fair cooperation was also exploited. The difference is that the work only provided an empirical study with no formal framework that justifies the optimal result. Also, the performance model and the distributed protocol are different.

Game theory is useful for addressing many different problems in edge computing, see [14] for a complete survey. In general, to force players to cooperate, pricing, reputation or some other external mechanism is to be used. For example, [15] studied a resource crowd-funding algorithm to integrate sporadic resources to form a dynamic resource pool that can optimize the spare resources in the local network. Here we assume that computation resources do not change over time. [16] presents a game-theory model for fog-edge cooperation. However, the goal of the protocol is to maximize a reward computed for each new task, whereas we consider as utility a performance metric. Finally, this problem has some analogy with file sharing in peer to peer (p2p) networking. However, the underlying performance model is different. For example, in p2p there is no notion of loads and quality of service. Also, the scale is much larger.

III. Computation resource sharing in the fog

We consider a set of N autonomous fog providers that wish to cooperate by sharing their computation resources. In the following the terms provider and node is used as synonymous. Before presenting our proposed protocol, we elaborate in general about the underlying nature of cooperation. The analysis is based on two ingredients: (i) a performance model; (ii) an formal problem definition, first in terms of game theory and then as an optimization problem.

The challenging aspect of cooperation is that as providers are competitors they may cheat when a computation resource is requested by another provider. Incentives may be used to cope with this issue. For example, in [12] we have studied a possible incentive mechanism based on virtual coins. However, when nodes have different loads, the frequencies of requests is not the same and the price of a resource has to be tuned to avoid that a node may experience a performance degradation due to a higher request rate; but, price regulation is not an easy task. Here we aim at studying a simpler mechanism that avoids coins or other incentives while ensuring an advantage under any load condition.

The proposed protocol is based on the power-of-two random choices mechanism. When a node needs a server, because of an overload condition, the node selects and probes one node of the group at random. To bring the selfish behavior of providers into the performance model, we add a degree of freedom to the classic Markov chain based analysis. More precisely, a provider i decides to cooperate and share one of its server with a probability p_i, called the *cooperation probability*.

A. Performance model

The performance model computes the *blocking probability* of tasks arriving to nodes, namely the probability that a task cannot be served by the node. We do not focus on delay. Nodes

have no waiting area, as in other cloud system models, [17] while the communication delay is negligible compared to the task execution time. Every node i has K servers and receives task execution requests according to Poisson process with rate λ_i requiring a unitary exponentially distributed service time.

The model of the system is an N-dimension birth-death process, $BD(N, K)$, with state space $[0, K]^N$. The state of the process, $\mathbf{k} = (k_1, k_2, \ldots, k_N)$, is a vector where k_i represents the number of busy servers of provider i. Let $\mathbf{e}_i$ be the vector of all zero, but 1 at position i. The transition rate $q(\mathbf{k}, \mathbf{k}')$ of the process can either be related to a birth event (a task that enters the system) or to a death one (a task that leaves the system). The birth rate is ($k_i < K$):

$$q(\mathbf{k}, \mathbf{k} + \mathbf{e_i}) = \lambda_i + \frac{1}{N-1} \sum_{\mathbf{k}:\mathbf{k}_j=K} \lambda_j p_i$$

The first term is the rate of tasks received by provider i while a term in the summation is due to a request coming from provider j when it is congested ($k_j = 1$), and i cooperates (this occurs with probability p_i). The dead rate is readily given by ($k_i > 0$):

$$q(\mathbf{k}, \mathbf{k} - \mathbf{e_i}) = k_i$$

A task arriving at node i is blocked when i is congested ($k_i = K$) and either the probed provider, say j, is congested as well ($k_j = K$), or it has available serves ($k_j < K$) but it cheats (this event occurs with probability $1 - p_j$). The blocking probabilities are then computed from the steady state probabilities of the BD process as follows:

$$b_i = \frac{1}{N-1} \sum_{k_i, k_j = K} \pi_{\mathbf{k}} + \frac{1}{N-1} \sum_{k_i = K, k_j < K} \pi_{\mathbf{k}} (1 - p_j)$$

B. Casting resource sharing as a game

Game theory is useful to analyze the behavior of distributed competing nodes, e.g., see [14]. In this section, we use a simple game-theory point of view to outline the key challenges behind cooperation. In the fog model, the approach aims at determining the utility of sharing a single resource, connected to some reward from clients. Here we tie the utility directly to the blocking probability performance metric, and the utility is to be interpreted as an expected value. More specifically, when casting the provider behaviors in the light of a game, we have to take into accounts that players interact in an asynchronous way since probe requests are neither alternating or simultaneous. Due to this random interactions among players, the game cannot use a classical model, i.e., where players move alternatively or simultaneously; rather, the game has to embraces these asynchronous moves. To use the usual game theory framework we consider average values as follows.

The strategy of player i is in general a mixed-strategy over two pure strategies: cooperate (played with probability p_i) or not cooperate. A mixed strategy is determined by p_i. We use as payoff $u_i = 1 - b_i$. The main problem behind cooperation is well outlined even with only two players.

		Node n_2	
		$p_2 = 0$	$p_2 = 1$
Node n_1	$p_1 = 0$	0.52,0.52	0.71,0.43
	$p_1 = 1$	0.43,0.71	0.63,0.63

Table I: The payoff matrix of a game for $\lambda_1 = \lambda_2 = 0.95$ and $K = 1$. An entry in the table contains (u_1, u_2). If both players cooperate, the utility of players is higher than when not cooperating.

		Node n_2	
		$p_2 = 0$	$p_2 = 1$
Node n_1	$p_1 = 0$	0.52,0.8	0.78,0.63
	$p_1 = 1$	0.51,0.92	0.67,0.67

Table II: The payoff matrix of the same game for $\lambda_1 = 0.95, \lambda_2 = 0.2$. The less loaded player gets a lower utility even if both players cooperate (0.67 instead that 0.8). If only one player cooperates, the utility of such a player is lower than without cooperation.

1) Prisoner's Dilemma: We start by considering a game with pure strategies, $p_i \in \{0, 1\}$, that corresponds to the case when providers can either decide to always cooperate or not cooperate and don't change the strategies thereafter. The $BD(2, K)$ process it used to compute the payoff table of this game. Table I reports a numerical example of the payoff matrix, where the values are computed for $K = 1, \lambda_1 = \lambda_2 = 0.95$. An entry in the table is a pair (u_1, u_2).

For *this* specific load profile, the payoff matrix is equivalent to the well known table of the Prisoner's Dilemma (PD). It is easy to see how the state 00 is a Nash equilibrium, namely none of the two players has incentive to unilaterally deviate from 00 given that the other player's strategy remains unchanged, as this unilateral deviation will not increase its utility. In particular, if a player cooperates and the other doesn't, its utility decreases from 0.52 to 0.43. The Nash equilibrium is not optimal, as 11 would provide a higher utility to both providers, 0.63 rather than 0.52. If the traffic is heavily asymmetric, i.e., $\lambda_1 \gg \lambda_2$, the game is no longer equivalent to the PD. For example, Table II is the payoff matrix for $\lambda_1 = 0.95, \lambda_2 = 0.2$. For this unbalanced case, the solution of the game 00 is also optimal. We will elaborate about the optimality of cooperation later in this section when switching to the problem optimization point of view.

2) Iterated Prisoner's Dilemma: Let now consider a dynamic game, i.e., a game where players can change their strategies. This game is organized in rounds. During a round a *stage* game is repeated with a fixed strategy and players can change the strategy at the end of a round. If the stage game is Table I, we get an instance of the Iterated Prisoner's Dilemma (IPD). It is known that if the IPD is played a known amount of times, then it is optimal to defect in all rounds. However, if the game is repeated infinitely, the Folk theorem ensures that the solution of the IPD is to cooperate.

3) General Iterated Game: By properly setting p_i, players can get benefit from cooperating even when loads are very different as in the first example. Before elaborating upon

		Node n_2		
		0	0.5	1
Node n_1	0	0.52,0.52	0.63,0.47	0.71,0.43
	0.5	0.47,0.63	**0.58,0.58**	**0.66,0.55**
	1	0.43,0.71	**0.55,0.66**	**0.63,0.63**

Table III: A numerical example of the payoff matrix for the previous two cooperating nodes, with 3 strategies. Bold numbers are payoff of possible solutions of the game.

this possibility, we study the consequence of allowing players to cooperate randomly in the stage game. Table III reports the Payoff matrix of stage a game where $p_i \in \{0, 0.5, 1\}$, $K = 1, \lambda_1 = \lambda_2 = 0.95$. In this case, all the the strategies $(0.5, 0.5), (1, 0.5), (0.5, 1), (1, 1)$ are possible equilibrium of an infinitely repeated game, as for each of these strategies $u_i > 0.52$. However, all strategies but (0.5,0.5) are Pareto-efficient. i.e., it is impossible to improve the utility of *both* players by changing even both the current strategies. Though iterating the game allows to escape from the not optimal Nash equilibrium, some additional mechanism is required to converge to a specific solution of the game. This is where fairness comes into the scene. To better illustrate this point we switch the analysis point of view.

C. Casting cooperation as an optimization problem

Optimal resource sharing can be expressed as the following optimization problem.

Problem $\mathcal{P}_{NK}(\lambda_1, \dots \lambda_N, K)$**:**

$$\underset{\mathbf{p}}{\text{Minimize:}} \quad b_1(\mathbf{p}), \dots, b_N(\mathbf{p}) \qquad (1)$$

Subject to:

$$b_i(\mathbf{p}) < b_i^0 \ \ \forall i = 1..n \quad (2)$$

$$0 \leq p_i \leq 1 \ \ \forall i = 1..n \quad (3)$$

Where the functions b_i in Equation (1) are the blocking probabilities obtained from $BD(N, K)$ and loads λ_i, wereas b_i^0 is the blocking probability of node i working without using any resource sharing. This value can be computed using the Erlang-B formula and it is equal to $b_i(\mathbf{0})$. The constraint (2) reflects the convenience for a node to cooperate.

1) The case of $N = 2$: The characterization of the solution of above problem for $N = 2$ can be done in a closed form for the case $K = 1$ and numerically for larger values of K. Recall that a solution $\mathbf{p}$ of a multi-objective optimization problem is Pareto-efficient iif it doesn't exit any other solution such that at least one b_i is worst than the one under $\mathbf{p}$. The set of Pareto-efficient solution is called the Pareto frontier.

Without loss of generality, assume that $\lambda_1 \geq \lambda_2$. We have the following

Theorem 1. *The Pareto-efficient solution set of the problem $\mathcal{P}_{21}$ is:*

$$\partial P = \{(1, p_2) | 0 < p_2^* \leq p_2 \leq 1\} \cup \{(p_1, 1) | 0 < p_1^* \leq p_1 \leq 1\}$$

if $\lambda_2 > \sqrt{\lambda_1 + 1} - 1$ and

$$\partial P = \{(1, p_2) | 0 < p_2 \leq p_2^* < 1\}$$

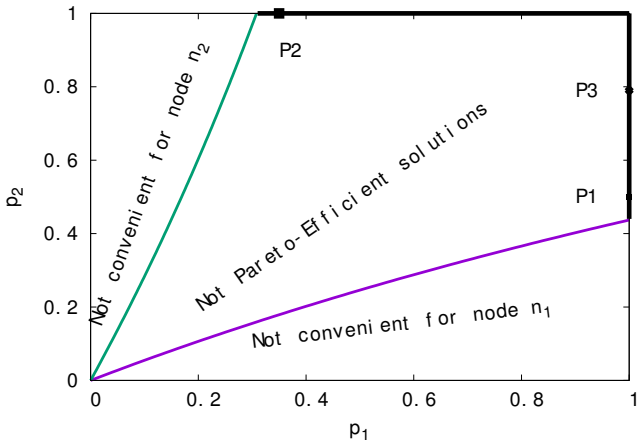

Figure 1: Partition of the domain of the variables for $\mathcal{P}_{21}$. Bold lines is the Pareto frontier. The three points are Pareto-efficient, but only P_3 is fair.

otherwise, where $p_1^*, p_2^* > 0$.

Proof. A sketch of the proof is given in the appendix. □

Discussion Figure 1 shows the domain $[0, 1]^2$ of $\mathcal{P}_{21}$ for $\lambda_1 = 0.9, \lambda_2 = 0.8$. The two lines are the isolines $l_i(p_1, p_2)$: $b_i(p_1, p_2) = b_i^0$, $i = 1, 2$, i.e., no improvement is achieved by node i if (p_1, p_2) belongs to isoline l_i. Points between the two isolines satisfies the constrain 2. The Pareto frontier is depicted in bold.

The solution landscape of the problem raises the question: how nodes agree to cooperate with any of the solution in the Pareto-frontier? For example, both points P_1, P_2 are optimal wrt to $\mathcal{P}_{21}$. However, under the node 1 perspective, it is clear that P_2 (where $p_1 < 1$) is more convenient than P_1 (where $p_1 = 1$). In fact, since in P_2 node 1 cooperates less frequently, the blocking probability b_1 is lower than the one corresponding to P_1. But, for the same reason, node 2 gets a higher improvement if the working point is P_1.

The following table reports b_i for these points (note that the utility of the game was defined as $1 - b_i$). The first row is the no cooperation case.

	(p_1, p_2)	b_1	b_2
–	(0,0)	0.47	0.44
P_1	(1,0.44)	0.44	0.30
P_2	(0.37,1)	0.31	0.45
P_3	(1,0.79)	0.38	0.33

We did several experiments with $N = 2$ and higher K that confirmed that the solution domains has the same structure of Figure 1. We conjecture that a similar property holds for any instance $\mathcal{P}_{NK}$

Conjecture For any $N \geq 2, K \geq 1$ and loads $\lambda_1, \dots, \lambda_N$,with $\lambda_1 \geq \lambda_i$, if $\mathbf{p}^* = (1, p_2^*, \dots, p_N^*)$ is admissible solution of $\mathcal{P}_{NK}$ then it is Pareto-optimal.

Intuitively, if we take the node 1 point of view, the node sees

all the other nodes as a single cooperating node with mean load $\frac{1}{N-1}\sum_{i>1}\lambda_i \geq \lambda_1$, and then Theorem 1 holds.

IV. Randomized fair cooperation among N nodes

In this section we first define the notion of fair cooperation among a set of nodes and then provide a numerical algorithm of determining the vector of cooperation probabilities that ensure fairness.

Definition 1 (Acceptance rate a_{ij})**.** *The task acceptance rate of node a node n_i wrt node j, a_{ij} is the rate at which n_i accepts and executes tasks sent by n_j to n_i.*

Definition 2 (Fair cooperation among N nodes)**.** *A set of N nodes cooperates fairly if*

$$\sum_{j\neq i} a_{ij} = \sum_{j\neq i} a_{ji} \quad \forall i = 1...N$$

Discussion Fair cooperation is an incentive mechanism to cooperate. As nodes are homogeneous fair cooperation boils down to borrow a server for an amount of time ΔT, and get a server back for the same amount of time ΔT. Hence, the blocking probabilities under fair cooperation cannot be worst than without cooperation since a node maintains its full average server capacity, and decreases with ΔT.

If all nodes agree on this criteria, then they have a common interest in maximizing the effect of resource sharing by increasing ΔT, i.e., increasing as much as possible their cooperation probabilities while meeting the fairness constrain. We now bring this notion in the problem and use it as the driving idea of our proposed protocol.

A. Determining fair cooperation probabilities

The vector of cooperation probabilities that ensures fair cooperation is found by adding the following constrain to $\mathcal{P}_{NK}$

$$\sum_{j\neq i} a_{ij} - \sum_{j\neq i} a_{ji} = 0 \quad i = 1,\ldots,N \tag{4}$$

This new problem is denoted as $\mathcal{P}^*_{NK}$. As we are adding a constrain to the original problem, the solutions of the constrained problem $\mathcal{P}^*_{NK}$ are a subset of the unconstrained problem, $\mathcal{P}_{NK}$. In particular, an efficient solution of $\mathcal{P}_{NK}$ that satisfies the constrain Equ. 4 is efficient solution for $\mathcal{P}^*_{NK}$. We now see that $\mathcal{P}^*_{NK}$ has a unique solution $\mathbf{p}^*$. For $N = 2, K = 1$ we have the following result.

Theorem 2 (Fair cooperation)**.** *The solution of $\mathcal{P}^*_{21}$ is*

$$p_1 = 1, p_2 = \left(\frac{\lambda_2}{\lambda_1}\right)^2$$

where $\lambda_1 \geq \lambda_2$.

Proof. See appendix. □

We now describe a numerical method to determine $\mathbf{p}^*$. First, from the BD model, the acceptance rates are:

$$a_{ij} = \frac{1}{N-1} p_j \sum_{\mathbf{k}:k_i=K,k_j<K} \lambda_i \pi_{\mathbf{k}}$$

Algorithm 1 Fixed Point Algorithm

```
p ← 1
Q ← Q(p)
π ← Solve[Qπ = 0, Σ π_s = 1]
F ← F(π)
while max(|Fp|) > ε do
    p ← Solve[Fx = [1|0]^T]
    Q ← Q(p)
    π ← Solve[Qπ = 0, Σ π_s = 1]
    F ← F(π)
end while
```

In fact, node j accepts a task from i when: the task arriving to node i finds the node congested and selected node j is not congested (this occurs with rate $\lambda_i \frac{1}{N-1}\pi_{\mathbf{k}}$) and node j accepts to execute the task (this occurs with probability p_j). Let

$$f_{ij} = \lambda_i \sum_{\mathbf{k}:k_i=K,k_j<K} \pi_{\mathbf{k}}$$

and

$$f_{ii} = -\sum_{j\neq i} f_{ji}$$

The constrains of Eq. 4 are expressed as (the denominator can be cancelled):

$$\begin{bmatrix} f_{11} & \cdots & f_{1j} & \cdots & f_{1N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ f_{i1} & \cdots & f_{ii} & \cdots & f_{iN} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ f_{N1} & \cdots & f_{Nj} & \cdots & f_{NN} \end{bmatrix} \times \begin{bmatrix} p_1 \\ \cdot \\ p_i \\ \cdot \\ p_N \end{bmatrix} = \mathbf{0}$$

More succinctly the vector $\mathbf{p}^*$ satisfies

$$\mathbf{F}\mathbf{p}^* = 0 \tag{5}$$

Matrix $\mathbf{F}$ is not singular as the sum of rows is zero:

$$\sum_i f_{ij} = \sum_{i\neq j} f_{ij} + f_{jj} = \sum_{i\neq j} f_{ij} - \sum_{j\neq i} f_{ij} = 0 \quad \forall j$$

The equation (5) and

$$\mathbf{Q}_{p^*}\pi = 0 \quad \sum_{\mathbf{k}} \pi_k = 1$$

is a system of $N+N^{K+1}$ equations in the N unknowns p_i and N^{K+1} unknowns $\pi_{\mathbf{k}}$ with one degree of freedom. According to our previous discussion, we bound $p_1 = 1$ and the solution $\mathbf{p}^* = (1, p_2, \ldots, p_N)$ of such a system of equations is unique and solves $\mathcal{P}^*_{NK}$.

1) Numerical algorithm: The vector $\mathbf{p}^*$ can be found by a direct method. However, due the exponential grow of $\mathbf{Q}$, the matrix inversion operation limits the size of the problem that can be solved. To overcome this limitation, we propose to use the fixed point algorithm (Alg. 1), where $\lambda_1 \geq \lambda_i, i > 1$ that breaks the solution in two parts. The algorithm first computes the state probabilities with the cooperation vector $\mathbf{1}$. Here, since the matrix $\mathbf{Q}$ is (very) sparse, BD can be solved via the well known power method thus avoiding inversion. Then, a matrix $\mathbf{F}$ is defined based on the solution $\pi_{\mathbf{k}}$. From here (line

6), a new vector $\mathbf{p}$ of cooperation probabilities is obtained by solving the following system:

$$\mathbf{F}'\mathbf{x} = [1|\mathbf{0}]^T,$$

where $\mathbf{F}'$ is obtained from $\mathbf{F}$ by replacing the first row with $(1, 0, \ldots, 0)$ so that $p_1 = 1$ (the size of this matrix is small so that it can be inverted). The vector $\mathbf{p}$ is then used to solve a new BD process. If the state probabilities found satisfy the fairness constrain (with some error ϵ), the vector provides fair cooperation. Otherwise, this procedure is repeated. The limit of this method is the number of states of BD.

B. Numerical results

We present here some result for $N = 4, 8, K = 1$ and a geometric load profile, i.e., $\frac{\lambda_i}{\lambda_{i+1}} = q$ [1]. The results are obtained with $\lambda_1 = 0.9$ and a profile is denoted with $G(q)$; *e.g.,* $G(1.05)$ means $\lambda_1 = 0.9, \lambda_2 = \frac{0.9}{1.05} \approx 0.875, \lambda_3 = \frac{0.875}{1.05} \approx 0.833$, and so on.

Load	n_1	n_2
G(1)	22.6	22.6
G(1.05)	21.3	23.5
G(1.1)	20.0	24.2
G(1.15)	18.9	25.0

Table IV: Cooperation gain, $N = 2$ nodes

Load	n_1	n_2	n_3	n_4
G(1)	25.4	25.4	25.4	25.4
G(1.05)	22.1	23.2	24.4	25.5
G(1.1)	19.3	21.4	23.5	25.5
G(1.15)	15.1	18.8	22.2	25.3

Table V: Cooperation gain, $N = 4$

Load	n_1	n_2	n_3	n_4	n_5	n_5	n_7	n_8
G(1)	26.3	26.3	26.3	26.3	26.3	26.3	26.3	26.3
G(1.05)	19.7	20.4	21.1	21.8	22.5	23.1	23.7	24.4
G(1.1)	15.2	16.4	17.6	18.7	19.8	20.8	21.7	22.6
G(1.15)	12.1	13.6	15.1	16.5	17.8	19.0	20.0	20.9

Table VI: Cooperation gain, $N = 8$

Tables IV, V, and VI report the percentage reduction in the blocking probabilities, referred to as the *cooperation gain* for different profiles and $N = 2, 4, 8$. First of all, for $q = 1$, i.e., equally loaded nodes, the gain is the same. The gain measured by the most loaded node n_1 is maximum when all nodes have the same loads, and decreases with q. For $N = 2$ this occurs because n_1, the most loaded node will probe n_2 more frequently than n_2 does. Hence, to ensure fairness $p_2 < p_1 = 1$, i.e., n_2 sees its task more frequently accepted and hence is gain is higher. This difference increases with q due to an increase in the traffic rates. For $N = 4$, the n_1's gain also decreases with q; the reason being that nodes n_2, n_3, n_4 are seen by n_1 as a single entity with an average load $\lambda_e = \frac{1}{3}(\lambda_2 + \lambda_3 + \lambda_4) = \frac{1}{3}(\frac{1}{q} + \frac{1}{q^2} + \frac{1}{q^2})\lambda_1 < \lambda_1$, which decreases with q. Hence, the previous analysis for $N = 2$ holds. Interestingly, for $N = 2$ the cooperation gain of n_1, n_2 for $q = 1.1$ is higher than when other nodes share their resource, i.e., $N = 4, N = 8$. This means that nodes n_1, n_2 get a higher advantage in cooperating only among themselves, i.e, forming a coalition. This convenience depends on the load. Finding the best node coalition among a set of nodes is left as a future work.

[1]We used $K = 1$ to reduce the time complexity of the fixed point algorithm. As discussed for $N = 2$ we guess that similar results are obtained with $K > 1$. This is corroborated by the simulation results discussed next.

V. DFR: Distributed Fair Randomized Cooperation

In this section, we propose a distributed algorithm running at the fog nodes to determine the optimal cooperation probability vector for each fog node. We first illustrate the idea of the algorithm through a centralized implementation. Then, we define the distributed version of the centralized algorithm.

A. Centralized Algorithm

The key idea behind our algorithm is that the solution of $\mathcal{P}^*_{NK}$ is equivalent to the problem of finding the zero of the following function:

$$g(\mathbf{p}) = \sum_{i=1}^{N} |1 - r_j(\mathbf{p})| \quad \mathbf{p} \in \{1\} \times [0, 1]^{N-1} \tag{6}$$

where

$$r_i = \frac{\sum_{j \neq i} a_{ij}}{\sum_{j \neq i} a_{ji}}$$

The basic idea of the algorithm is to find the zero of g by sequentially selecting a suitable j and finding the cooperation probabilities that makes $r_j \sim 1$ (Alg. 2). This operation is repeated until all $r_j \sim 1$ or a maximum number of iterations is reached.

(a) *how to select j?*: The index j is selected in a way that the $r_j \sim 1$ can be reached by only changing p_j and keeping the other p_i fixed. This property will allow to implement the algorithm in a distributed way (see later). To ensure that this can be obtained, j must be such that $r_j > 1$. In fact, suppose that for a given $\mathbf{p}$, with $p_j = p'_j > 0$, $r_j(\mathbf{p}) > 1$. From it definition if $p_j = 0$ then $a_{ij} = 0$ (the node doesn't accept any task) and hence $r_j = 0$.

Now, it is reasonable to assume that r_j is a monotonic function of p_j, so that a unique value $p^*_j, p_j < p^*_j < 0$ such that $r_j = 1$ should exist. The algorithm selects the node with maximum $r_j > 1$ (lines 3 and line 7).

(b) *how to find p^*_j?*: To compute p^*_j we leverage a bisection algorithm for its robustness compared to convergence.

Alg. 2 first computes the cooperation ratios r_j for $\mathbf{p} = \mathbf{1}$. It then selects the node j with the maximum $r_j > 1 + \epsilon$, where ϵ is an allowed approximation error, and call a bisection search algorithm on the selected node j. These steps are repeated until no nodes can be selected or the maximum allowed number of cycles is reached. Figure 2 shows a trace of the centralized algorithm.

Algorithm 2 DFR centralized

Input: $Steps, \epsilon$
Output: $\mathbf{p}$
$\mathbf{p} \leftarrow \mathbf{1}$
$\mathbf{r} \leftarrow$ Compute_Ratios($\mathbf{p}$)
$j \leftarrow \text{argmax}_i \{r_i \geq 1 + \epsilon\}$
while $j \neq null \ \& \ Step > 0$ **do**
  $\mathbf{p}_j \leftarrow$ Bisection(j)
  $\mathbf{r} \leftarrow$ Compute_Ratios($\mathbf{p}$)
  $j \leftarrow \text{argmax}_i \{r_i \geq 1 + \epsilon\}$
  $Step \leftarrow Step - 1$
end while

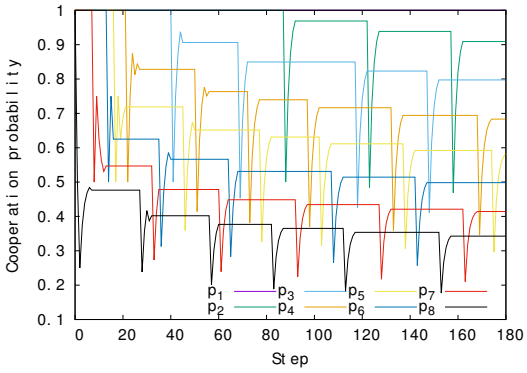

Figure 2: Example of cooperation probability tuning, $\lambda_1 = 0.9$. The other loads are $\lambda_{i+1} - \lambda_i = 0.1$ (4 nodes) or $\lambda_{i+1} - \lambda_i = 0.05$ (8 nodes).

B. DFR: Distributed Implementation

We propose a candidate distributed protocol that tunes the cooperation probabilities based on the previous centralized algorithm. The protocol takes the following two considerations into accounts: (*i*) when running the centralized algorithm we observed that the order with which nodes is selected is maintained over the iterationsand that this order reflects their loads (the values r_j are inversely proportional to the loads). The distributed protocol exploits this observation to implement a token circulating on a logical ring formed at the beginning the algorithm, where nodes are ordered according to their r ; (*ii*) the cooperation ratio is an approximation of the ideal ratio, hence subject to noise with false positives and negatives. To ensure that the bisection algorithm terminates a condition on the length of the current interval $[max - min]$ is added.

Our distributed algorithm, depicted in Alg. 3, operates as follows. We assume that nodes are connected and can communicate with each other. The algorithm is triggered by any node initiator, say u. An initiator node u triggers the algorithm if: (*i*) u decides to cooperate in order to reduce its blocking probability (if nodes do not cooperate), or (*ii*) u decides to change the current cooperation probabilities (*e.g.,* if it detects a change in the current cooperation ratio). The algorithm consists of a *warm-up* phase and R *tuning rounds*. Node u sends a broadcast (*bcast*) message to all the other nodes indicating a warm-up phase of the cooperation algorithm is going to be started. This phase lasts for a given time, ΔT, specified in the message.

Any node j, receiving this message, sets its cooperation probability to $p_j = 1$ for the ΔT time interval. During this interval, j measures the number of tasks received from other nodes for local execution at j, in_j and the number of tasks that j has sent for remote execution at other fog nodes $j' \neq j$, out_j. At the end of this warm-up phase, every node j sends a reply message carrying $r_j = \frac{in_j}{out_j}$ to u. After receiving all the replies from the nodes, node u creates a sorted list s of nodes according to their values r_j, with the top ranked being the node with the highest r_j. Node u then creates a token message with this sorted list and an integer R. The message loops through the ring of nodes following the order in s, performing R rounds. The token represents the right of a node to change its cooperation probability using a bisection algorithm. To estimate r_j in the future, node j uses two counters, in_j and out_j. The value $r_j = \frac{in_j}{out_j}$ is computed soon as either in_j or out_j is equal to k, where k is a protocol parameter. The owner of the token starts using $max_j = 1, min_i = 0$ and tunes its cooperation probability until either r_j is (close to) 1, or $max_j - min_j < \epsilon$. This stopping condition is defined to take the possible estimation errors of r into account.

Algorithm 3 DFR Protocol running on Node j

On accepting a Task:
$in_j \leftarrow in_j + 1$
if $in_j == k$ **then**
  Tune()
end if
On sending a Task:
$out_j \leftarrow out_j + 1$
if $out_j == k$ **then**
  Tune()
end if
On Tune:
$r_j \leftarrow \frac{in_j}{out_j}$
$in_j, out_j \leftarrow 0$
if $(r_j \geq 1 + \epsilon) \ \& \ (max_j - min_j > \epsilon)$ **then**
  $max_j \leftarrow p_j$
  $p_j \leftarrow (max_j + min_j)\frac{1}{2}$
  return
end if
if $r_j \leq 1 - \epsilon \ \& \ max_j - min_j > \epsilon$ **then**
  $min_j \leftarrow p_j$
  $p_j \leftarrow (max_j + min_j)\frac{1}{2}$
  return
end if
if not.Token.last **then**
  $update\_Token()$
  Send_To_Next(Token)
end if

1) Result: Fig. 3 shows the results of the simulation using our distributed algorithm running on a $N = 4$ node network. We vary the number of servers per node and plot the blocking probability over time for $K = 10$ servers. We set the Fog-to-Fog link communication rate at 54 Mpbs and the latency uniformly distributed in $U[0.5, 1.2]$. The distributed algorithm converges in 50 steps. Due to space limitation we do not report other simulation results. Overall, our study provided evidence that the algorithm converges for different load profiles servers and number of nodes.

VI. Conclusion

This paper has investigated fog-to-fog cooperation within a multi-providers multi-stakeholders scenario. We have proposed

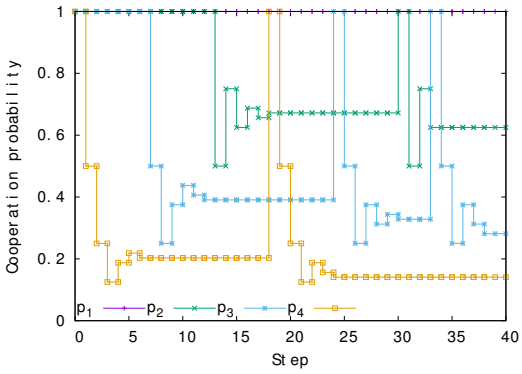

Figure 3: A trace of the cooperation protocol tuning phase for $N = 4$ nodes, $K = 10$ servers, $\lambda_1 = 0.9, \lambda_2 = 0.8, \lambda_3 = 0.7 \lambda_4 = 0.6, k = 5000, \epsilon = \delta P = 10^{-2}$

a Markov Chain based model and formulate the F2F cooperation as a minimization problem under fairness constraints and assuming uniform cooperative fog node selection. The problem has been solved in a closed form for $N = 2$ and numerically for arbitrary N fog nodes. A distributed protocol for probability cooperation tuning has been proposed and simulation results of the protocol have shown the major gain of F2F cooperation when compared to traditional no cooperating fog architectures.

APPENDIX

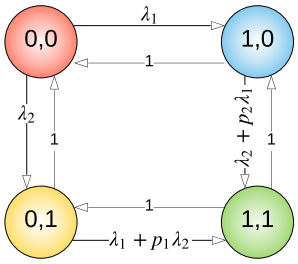

Figure 4: The state transition diagram of $BD(2, 1)$.

Theorem 1

Proof. (sketch) The $BD(2, 1)$ process is represented as a two dimensions Markov Chain (2-MC) in Figure 4. The $\mathbf{Q}$ matrix of the chain is

$$\begin{bmatrix} -\lambda_1 - \lambda_2 & \lambda_1 & \lambda_2 & 0 \\ 1 & -1 - p_2\lambda_1 - \lambda_2 & 0 & \lambda_2 + p_2\lambda_1 \\ 1 & 0 & -1 - p_1\lambda_2 - \lambda_1 & \lambda_1 + p_1\lambda_2 \\ 0 & 1 & 1 & -2 \end{bmatrix}$$

The blocking probability b_i is then expressed as a multivariate function of p_1, p_2.

To proof the claim one has to show that $\frac{\partial b_1}{\partial p_1} >, \frac{\partial b_2}{\partial p_2} > 0$, as well as that it exits a vector $\delta\mathbf{p} = (\delta p_1, \delta p_2), \delta p_i > 0$ such that $\nabla b_1 \cdot \delta\mathbf{p} < 0, \nabla b_2 \cdot \delta\mathbf{p} < 0$, i.e., for a given solution vector $\mathbf{p}$, an increase one in both probabilities that reduces *both* b_i exits, while if only p_i is varied, b_i increases. Hence, any internal point of $[0, 1]^2$ is not Pareto-efficient and only the solutions with at least one component $p_i = 1$ cannot be improved and are then Pareto-efficient. The proof is completed after showing that when a $p_i = 1$, a value $p_j^* > 0$ of the other component such $b_i < b_i(0, 0)$ exists. □

Theorem 2

Proof. (sketch) It is enough to show that this pair is solution of the previous 2-MC and that the acceptance rates of the two nodes are the same, $a_{12} = a_{21}$. □

ACKNOWLEDGEMENT

Authors would like to thank Abderrahmen Mtibaa for his suggestions in describing the algorithms.

REFERENCES

[1] M. Satyanarayanan, W. Gao, and B. Lucia, "The computing landscape of the 21st century," in *HotMobile*, 2019.
[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *Internet of Things Journal*, vol. 3, October 2016.
[3] A. Mtibaa, K. A. Harras, and A. Fahim, "Towards computational offloading in mobile device clouds," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1. IEEE, 2013, pp. 331–338.
[4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.
[5] O. C. A. W. Group, "Openfog reference architecture for fog computing," in *https://www.openfogconsortium.org/ra/*. OpenFog Consortium, 2017.
[6] ETSI, "Multi-access edge computing (mec); framework and reference architecture mec 003 v2.1.1," 2019.
[7] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," in *IEEE Communications Surveys & Tutorials*. IEEE, 2018.
[8] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1094–1104, 2001.
[9] vY. Xiao and M. Krunz, "Qoe and power efficiency tradeoff for fog computing networks with fog node cooperation," *2017 Proceedings of IEEE INFOCOM*, 2017.
[10] W. Masri, I. A. Ridhawi, N. Mostafa, and P. Pourghomi, "Minimizing delay in iot systems through collaborative fog-to-fog (f2f) communication," *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 1005–1010, 2017.
[11] A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z. Patrikakis, "A cooperative fog approach for effective workload balancing," *IEEE Cloud Computing*, vol. 4, pp. 36–45, 2017.
[12] R. Beraldi, A. Mtibaa, and A. N. Mian, "CICO: a credit-based incentive mechanism for COoperative fog computing paradigms," in *2018 IEEE Global Communications Conference: Selected Areas in Communications: Internet of Things (Globecom2018 SAC IoT)*, Abu Dhabi, United Arab Emirates, Dec. 2018.
[13] H. A. Roberto Beraldi, Abderrahmen Mtibaa, "A power-of-two choices based algorithm for cooperation in fog computing under different loads," *IEEE Transaction on Cloud Computing*, 2018 Early Access.
[14] J. Moura and D. Hutchison, "Game theory for multi-access edge computing: Survey, use cases, and future trends," 2018.
[15] Y. Sun and N. Zhang, "A resource-sharing model based on a repeated game in fog computing," *Saudi Journal of Biological Sciences*, vol. 24, no. 3, pp. 687 – 694, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319562X17300529
[16] Z. Cao, H. Zhang, B. Liu, and B. Sheng, "A game-theoretic framework for revenue sharing in edge-cloud computing system," *CoRR*, vol. abs/1711.10102, 2017.
[17] Y. L. Qiaomin Xie, Xiaobo Dong and R. Srikant, "Power of d choices for large-scale bin packing: A loss model," in *Proceedings of ACM SIGMETRICS 2015*. IEEE, 2015.