Flocking Virtual Machines in Quest for Responsive IoT Cloud Services

Sherif Abdelwahab and Bechir Hamdaoui
School of EECS, Oregon State University
{abdelwas,hamdaoui}@eecs.oregonstate.edu

***Abstract*—We propose Flock; a simple and scalable protocol that enables live migration of Virtual Machines (VMs) across heterogeneous edge and conventional cloud platforms to improve the responsiveness of cloud services. Flock is designed with properties that are suitable for the use cases of the Internet of Things (IoT). We describe the properties of regularized latency measurements that Flock can use for asynchronous and autonomous migration decisions. Such decisions allow communicating VMs to follow a *flocking-like* behavior that consists of three simple rules: separation, alignment, and cohesion. Using game theory, we derive analytical bounds on Flock's Price of Anarchy (PoA), and prove that flocking VMs converge to a Nash Equilibrium while settling in the best possible cloud platforms. We verify the effectiveness of Flock through simulations and discuss how its generic objective can simply be tweaked to achieve other objectives, such as cloud load balancing and energy consumption minimization.**

***Index Terms*—Internet of things, Edge computing, Game theory, Resource management.**

I. Introduction

Live Virtual Machine (VM) migration is essential for improving the responsiveness of cloud services. VMs, hosted in Edge Cloud (EC) platforms, install resource-rich cloud services close to data sources to realize Internet of Things (IoT) applications. Such VMs provide, for example, real-time video analytics services from cameras to mobile applications. Hosting VMs near the edge can subdue the end-to-end services latency from hundreds to tens of milliseconds compared to hosting VMs in conventional cloud platforms [1], [2]. Existing resource provisioning algorithms migrate VMs from one EC to the other in response to devices mobility and services computational capacity requirements. VM migration ensures minimal average latency between mobile devices and the EC [3], [4]. Migration can achieve other goals such as load balancing, efficient service chaining and orchestration, infrastructure cost minimization, efficient content delivery, and energy efficiency, to name a few [5].

However, migrating VMs in response to the mobility of IoT devices is a limited decision given the diverse IoT use cases. Such a mobility-triggered migration assumes a constraining IoT use case in which a cloud service, installed in VMs, communicates with one - and only one - IoT device [2]. Consider EC services for smart glasses as an example. Migrating a VM, which hosts video processing cloud services of a Google glass, minimizes the video processing latency as the glass/user moves, a goal that is only reasonable for the Google Glass use case, in which the Google Glass is the singleton client that uses the EC video processing services [2], [3]. However, in general IoT use cases, several VMs, very likely to be deployed in different ECs, are needed to execute distributed algorithms (e.g. computer vision feature extraction, consensus, and aggregation [6]) and require intensive communication among the VMs and/or the devices. Distributed computer vision, for example, enables distributed context-aware applications such as autonomous vehicles and intelligent traffic systems [7]. Predominantly for modern IoT applications, developers implement analytics via large-scale distributed algorithms executed by VMs in geographically distributed and heterogeneous cloud platforms [8], [9], [10].

In this paper, we propose Flock; a simple protocol by which VMs autonomously migrate between heterogeneous cloud platforms to minimize their weighted latency measured with end-user applications, IoT devices, and other peer-VMs. In Flock, a VM uses only local latency measurements, processing latency of its hosting cloud, and local information provided by its hosting cloud about other clouds which the VM can migrate to. A VM greedily migrates to a cloud platform that reduces its regularized weighted latency. We prove, using game-theory, that Flock converges to a Nash Equilibrium (NE) and its Price of Anarchy (PoA) is $(1+\epsilon)$ given the properties of our proposed social value function. We discuss how Flock serves a generic goal that we can redesign using simple tweaks to achieve other objectives such as load balancing and energy minimization. Our design allows VMs to imitate a *flocking-like* behavior in birds comprising separation, alignment, and cohesion rules.

II. System Model and Objective

We consider a network of VMs modeled as a graph $G = (V, P)$, where V denotes the set of n VMs and P denotes the set of VM pairs such that $p = (i, j) \in P$ if the i-th and j-th VMs communicate with each other. Let $d_{ij} \in \mathbb{R}^+$ denote the traffic demand between VMs i and j and assume that $d_{ij} = d_{ji}$. We also consider a set, A, of m clouds (i.e. ECs, or conventional clouds) that communicate over the Internet. A VM i autonomously chooses its hosting cloud.

Let $x_i \in A$ denote the cloud that hosts i and let $l(x_i, x_j) > 0$ be the average latency between i and j if they are hosted at x_i and x_j respectively (Note: if i and j are hosted at the same cloud $x_i = x_j$). We assume that l is reciprocal and monotonic. Therefore, $l(x_i, x_j) = l(x_j, x_i)$ and there is an entirely nondecreasing order of $A \rightarrow A'$ such that for any consecutive $x_i, x_i' \in A'$, $l(x_i, x_j) \leq l(x_i', x_j)$. The reciprocity condition ensures that measured latencies are aligned with peer-VMs and imitates the alignment rule in bird flocking. We model $l(x_i, x_j) = \tau(x_i, x_j) + \rho(x_i) + \rho(x_j)$, where $\tau(x_i, x_j)$ is the

average packet latency between x_i and x_j, and $\tau(x_i, x_j) = \tau(x_j, x_i)$. The quantity $\rho(x)$ is the average processing delay of x modeled as: $\rho(x) = \delta \sum_{i \in V: x_i = x} \sum_{j \in V} d_{ij} / (\gamma(x) - \sum_{i \in V: x_i = x} \sum_{j \in V} d_{ij})$, where δ is an arbitrary delay constant and $\gamma(x)$ denotes the capacity of x to handle all demanded traffic of its hosted VMs. An increased value of $\rho(x_i)$ signals the VM i that it is crowding with other VMs in the same cloud, imitating the separation rule in bird flocking.

A VM i evaluates its weighted latency with its peers if hosted at x as

$$u_i(x) = \sum_{j \in V} d_{ij} l(x, x_j) / \sum_{j \in V} d_{ij}. \quad (1)$$

Our objective is to design an autonomous VM migration protocol that converges to an outcome $\sigma = (x_1, x_2, \ldots, x_n)$ that minimizes the sum of weighted latency $\sum_{i \in V} u_i(x_i)$. That is to say, σ maximizes the responsiveness of all VMs given their peer-to-peer communication pattern (i.e. connectivity and demands).

III. Flock: Autonomous VM Migration Protocol

We design a simple protocol that allows a VM to autonomously decide its hosting cloud among a set $A_i \subseteq A$ of available clouds by relying on only local regularized latency measurements. We call A_i the strategy set of i. Every cloud x evaluates its current weight $w_x = \sum_{i: x_i = x} u_i(x)$ and advertises a monotonic non-negative regularization function $f(w_x) : \mathbb{R}^+ \to \mathbb{R}^+$, such that $\alpha < f(w_x) < 1$ for $\alpha > 0$, to all the VMs that are hosted at x.

Since each VM is hosted by a single cloud and all VMs have access to the same strategy set, the VM migration problem is modelled as a singleton symmetric weighted congestion game with the objective of minimizing the social cost $C(\sigma) = \sum_{x \in A} w_x f(w_x)$. If $f(w_x)$ is approximately 1, this game model is approximately equivalent to minimizing $\sum_{i \in V} u_i(x_i)$. Throughout, we use $x \xrightarrow{i} y$ to mean that a VM i migrates from cloud x to cloud y. Letting $\eta \leq 1$ be a design threshold, we propose the following migration protocol:

Flock: Autonomous VM migration protocol.

Initialization: Each VM $i \in V$ runs at a cloud $x \in A$.
Ensure: A Nash equilibrium outcome σ.

1: During round k, do in parallel $\forall i \in V$
Greedy migration process imitating cohesion in flocking:
2: i solicits its current strategy A_i from x.
3: i randomly selects a target cloud $y \in A_i$.
4: **if** $u_i(y) f(w_y + u_i(y)) \leq \eta u_i(x) f(w_x - u_i(x))$ **then**
5: $x \xrightarrow{i} y$.
6: **end if**

We begin by proving that Flock converges to a Nash equilibrium, where each VM chooses a single cloud (strategy) and no VM has an incentive (i.e. less average latency) to migrate from its current cloud. Then, we derive an upper bound on Flock's PoA for general regularization functions, f, following a similar approach to [11]. Finally, we propose a regularization function $f(w_x) \approx 1$ that achieves a tight PoA of at most $1 + \epsilon$.

A. Equilibrium

Convergence to a Nash equilibrium in Flock is non-obvious given the inter-dependency of a VM's average weight with its peers.

Theorem 3.1: Flock converges to a Nash equilibrium outcome.

Proof Any step of Flock corresponds to choosing a random outcome from a finite number of outcomes. We show that any step of Flock reduces the social value $C(\sigma)$ and $C(\sigma)$ is bounded below. We first show that if a VM i migrates from cloud x to cloud y, the increase in y's weight is less than the decrease in x's weight. We then use contraction to show that if a subset of i's peers have a total latency that increased after i migrates, the remaining peers must have a total latency that decreased by a greater value. Let σ^k and w_x^k denote the game outcome and the weight of x at round k respectively, and let $\Delta w_x = w_x^{k+1} - w_x^k$. If $x \xrightarrow{i} y$, then

$$u_i(y) f(w_y + u_i(y)) \leq \eta u_i(x) f(w_x - u_i(x)).$$

As $\alpha < f < 1$, we have two extreme cases: $u_i(y)\alpha \leq \eta u_i(x)$, or $u_i(y) \leq \eta \alpha u_i(x)$, which implies that:

$$u_i(y) \leq \frac{\eta}{\alpha} u_i(x). \quad (2)$$

Because of the migration: $\Delta w_x < 0$ and $\Delta w_y > 0$, but $|\Delta w_x| \geq u_i(x)$, and $|\Delta w_y| \leq u_i(y)$, otherwise i would not migrate, then

$$|\Delta w_y| \leq |\Delta w_x|. \quad (3)$$

Let $z_i = u_i(x_i) f(w_{x_i})$ denote the regularized weighted latency and let z_i^t denote its value at round k. After the migration, i's peers split into two subsets: $V_{\text{inc}} = \{j \in V : z_j^{k+1} > z_j^k\}$, and $V_{\text{dec}} = \{j \in V : z_j^{k+1} < z_j^k\}$. Assume after the migration step that $\sum_{j \in V_{\text{inc}}} z_j > \sum_{k \in V_{\text{dec}}} z_k$. For $f \approx 1$, $\sum_{j \in V_{\text{inc}}} u_j(x_j) > \sum_{k \in V_{\text{dec}}} u_k(x_k)$. Substitute with the weighted latency value from (1) and since the demands (i.e. d_{ij}) remain unchanged, $\sum_{j \in V_{\text{inc}}} l(x_j, y) > \sum_{k \in V_{\text{dec}}} l(x_k, y)$. By the reciprocity of l, $\sum_{j \in V_{\text{inc}}} l(y, x_j) > \sum_{k \in V_{\text{dec}}} l(y, x_k)$, and $u_i(y) \geq u_i(x)$, which contradicts (2). By this contradiction and from (3), the social value $C(\sigma^{k+1}) \leq C(\sigma^k)$. Since n and m are finite, the number of all possible outcomes is finite. An outcome after a VM migration, σ^{k+1}, corresponds to randomly choosing an outcome from the finite outcome space which reduces the social value. Since $C(\sigma) > 0$, then Flock must converge to a Nash equilibrium. ■

B. Price of Anarchy

We first give a generic upper bound of the PoA, then we propose our regularization function that tightens the PoA.

Lemma 3.2: The social value of Flock has a perfect PoA at most $\lambda/(1-\varepsilon)$ if for $\varepsilon < 1$ and $\lambda > 1 - \varepsilon$ the regularization function satisfies $w^* f(w + w^*) \leq \lambda w^* f(w^*) + \varepsilon w f(w)$, where $w \geq 0$ and $w^* > 0$.

Proof Let σ denote a Nash outcome and σ^* denote any alternative outcome. Also let w_x and w_x^* denote the weight on x in outcomes σ and σ^* respectively. Similarly, let x_i and x_i^* denote i's strategy (hosting cloud) in σ and σ^* respectively. By definition of a Nash outcome, $\forall i,\ u_i(x_i) f(w_{x_i}) \leq u_i(x_i^*) f(w_{x_i^*})$.

Summing over all VMs we get, $C(\sigma) = \sum_{i \in V} u_i(x_i) f(w_{x_i}) \leq \sum_{i \in V} u_i(x_i^*) f(w_{x_i^*})$ However,

$$\begin{aligned}\sum_{i \in V} u_i(x_i^*) f(w_{x_i^*}) &\leq \sum_{x \in A} \sum_{i: x_i = x} u_i(x) f(w_x + u_i(x)) \\ &\leq \sum_{x \in A} w_x^* f(w_x + w_x^*) \\ &\leq \sum_{x \in A} \lambda w_x^* f(w_x^*) + \varepsilon w_x f(w_x) \\ &= \lambda C(\sigma^*) + \varepsilon C(\sigma).\end{aligned}$$

Then, $C(\sigma) \leq \lambda C(\sigma^*) + \varepsilon C(\sigma)$. Rearranging we get, POA $= C(\sigma)/C(\sigma^*) \leq \lambda/(1-\varepsilon)$. ■

Theorem 3.3: The regularization function $f(w) = exp(-1/(w+a))$ tightens the PoA to $1+\epsilon$ for a sufficiently large constant a and reduces the game to the original VM migration problem, i.e. minimizing $\sum_i u_i(x_i)$.

Proof For

$$\lambda \leq \frac{f(w_{\max} + w_{\min})}{f(w_{\min})} \left(1 - \varepsilon \frac{w_{\max} f(w_{\max})}{w_{\min} f(w_{\max} + w_{\min})}\right),$$

$f(w) = \exp(-1/(w+a))$ satisfies the condition $w^* f(w + w^*) \leq \lambda w^* f(w^*) + \varepsilon w f(w)$ (verify by inspection). For an infinitesimally small value of ε, we can choose a such that $\lambda = 1 + \epsilon$, hence the POA $\leq 1 + \epsilon$. If a is sufficiently large $f(w_x) \approx 1$, hence $C(\sigma) \approx \sum_i u_i(x_i)$. ■

IV. Experimental Results and Discussion

We simulate Flock using SimPy (see [12] with several simulation rounds. In each round we describe the clouds as a complete graph data structure with inter-cloud latency modeled as $\tau \sim$ Uniform$(10, 100)$ and cloud capacity as $\gamma \sim$ Uniform$(50, 100)$. We simulate the peer-to-peer relations of VMs as a binomial graph with $d \sim$ Uniform$(1, 10)$. Each simulated VM asynchronously runs Flock.

Fig. 1 shows the average rounds, k, required to converge to a Nash equilibrium at 95%-confidence interval with 0.1 error in simulations. Although in the worst case $k = O(n \log(n f_{max}))$, where f_{max} is the maximum value of the regularization function f [13], Fig. 1 suggests that Flock scales better than $O(n)$ on average. The proof of the average convergence time is complex and depends on properties of the social value $C(\sigma)$ and other parameters. We leave this proof for future work.

Fig. 2 shows the PoA statistics of Flock with different η. We implemented the optimal solution as brute-force in simulations that evaluates the minimum social value among all possible VM to cloud assignments. As $\eta \approx 1$, the maximum PoA achieved matched the theoretical value of 1.21. In practice, it is desirable to keep $\eta < 1$ (e.g. $\eta = 0.7$) to avoid migrations with insignificant improvements and alternating migrations between clouds. Although the worst case PoA for $\eta = 0.7$ approaches 4.5, the average PoA is acceptable in practice.

Fig. 3 shows the fairness of Flock as it maintains a homogenous PoA for individual VM. In this analysis, we evaluate the minimum utility value of each VM and compare it to the utility value achieved by Flock. As shown, Flock maintains a homogenous PoA accross individual VMs.

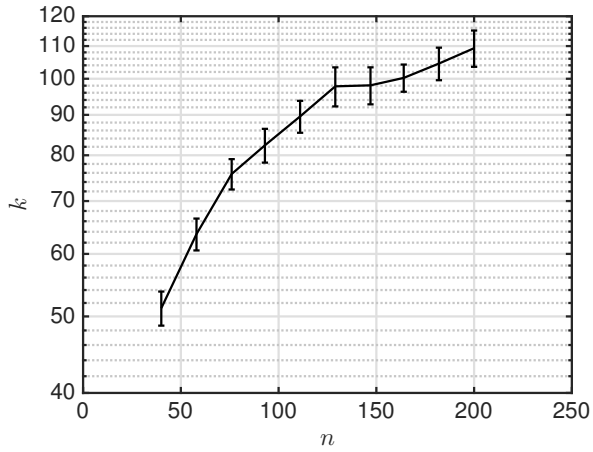

Fig. 1: Convergence of Flock: $m = 37$, $\tau \sim$ Uniform$(10, 100)$, $d \sim$ Uniform$(1, 10)$, $a = 9$, $\gamma \sim$ Uniform$(50, 100)$, and $\eta = 0.9$.

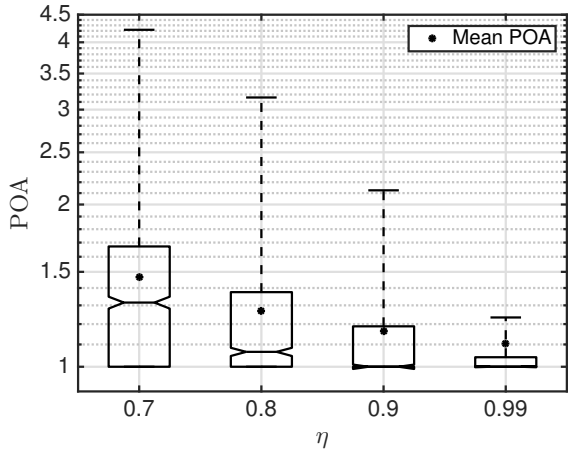

Fig. 2: Price of Anarchy statistics for $m = 5$, $n = 8$, $\tau \sim$ Uniform$(10, 100)$, $d \sim$ Uniform$(1, 10)$, $\gamma \sim$ Uniform$(50, 100)$, and $a = 9$.

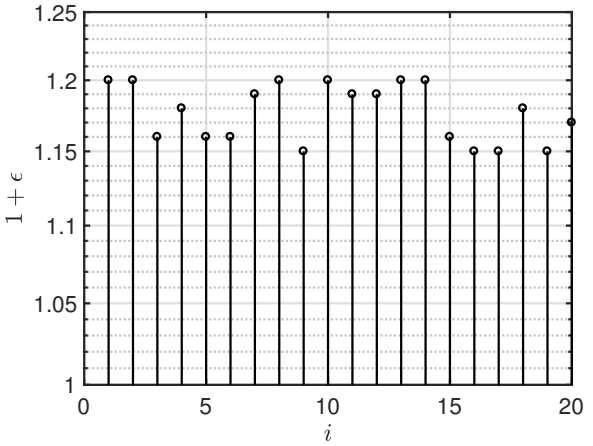

Fig. 3: Price of Anarchy statistics for individual VMs (i), $n = 8$, $\tau \sim$ Uniform$(10, 100)$, $d \sim$ Uniform$(1, 10)$, $\gamma \sim$ Uniform$(50, 100)$, and $a = 9$.

A. Special Cases: Load-balancing and Energy Efficiency

Flock can be redesigned with simple tweaks to suit other common cloud resource provisioning problems. We consider load-balancing and energy efficiency as special cases.

In load balancing, we seek an outcome σ such that the total load is distributed proportionally to the clouds' capacities. We first force the latency $\tau = 0$ between any cloud pairs $x, y \in A$. This is equivalent to marginalizing the effect of latency on the social value. We also force the VMs to ignore their peer relationships (i.e. $P = \emptyset$). The problem transforms immediately to minimizing the sum of cloud utilization. A VM in this setup greedily migrates to the least loaded cloud.

Fig. 4 shows the ideal mean utilization for a load-balanced system ('+') and the mean utilization using Flock ('*'). The box-plot also gives a summary statistics of the extreme value and the 75%-percentile samples. Flock performs very well as a load balancing protocol.

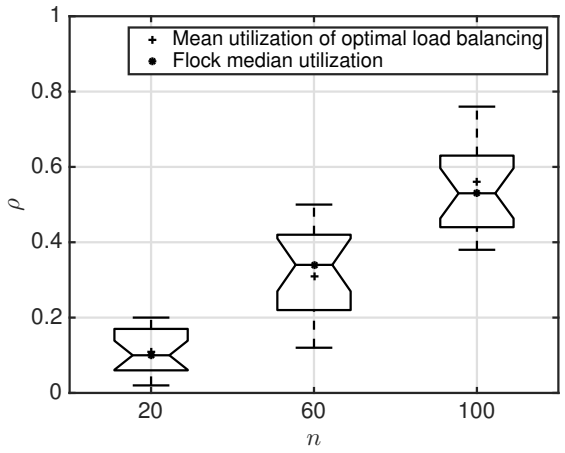

Fig. 4: Flock usage for load balancing ($P = \emptyset$, $\tau = 0$) for $m = 20$, $d \sim \text{Uniform}(1, 10)$, and $\gamma \sim \text{Uniform}(50, 100)$.

The energy-efficiency goal seeks an outcome σ in which the maximum number of clouds are idle (i.e. with $\rho = 0$). We (virtually) force each VM to be connected to all other VMs (i.e. $P = V \times V$). We also force τ to be a very large value (i.e. $\tau \to \infty$, and $d = 1$). With this tweak, a VM favors a cloud that hosts the largest number of VMs and with enough capacity ρ. Fig. 5 shows the ideal number of idle clouds that minimizes the energy consumption under a certain load (upper bound) and the number of idle clouds using Flock under the same load.

B. Impact of system dynamics

We now study the impact of various system dynamics on Flock's convergence. If we considered any two Flock steps at discrete times $(k-1)$ and k, the values of the VM's utilities, u_i, evolve randomly. Both external factors and VM migrations can influence this evolution and introduce difficulty in the analysis of Flock convergence. The simplest approach to deal with such dynamics is to assume that changes in latencies, EC capacities, and any other state are much slower than Flock convergence. This means that changes in u_i due to external factors such as link bandwidths or EC failures varies slowly. It also means that

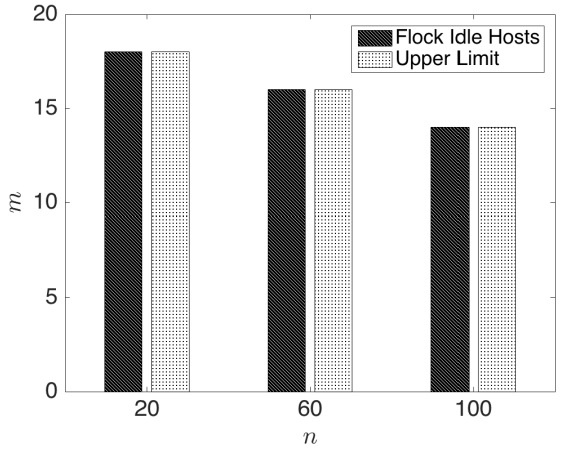

Fig. 5: Flock usage for energy efficiency ($P = V \times V$, $\tau \to \infty$, $d = 1$) for $m = 20$, and $\gamma \sim \text{Uniform}(50, 100)$.

Flock steps are very rapid and Flock measures only average values of l or any other system state. Fortunately, we have shown that Flock converges in a finite number of steps. However, we cannot guarantee that this number of steps are taken much faster than changes in any system state. We will follow the approach in [14] to show that despite that Flock may take slow migration decisions, its convergence still holds if we modeled the network dynamics (e.g. l) as a continuous-time Markov process that has values that are generated by the migration steps of Flock.

Consider modeling the utility of a VM as a general discrete-time stochastic process such that: $\forall k \in \mathbb{Z}, u_{i,k+1} = u_{i,k} + b_k \omega_k$, where $u_{i,k+1}$ is the utility value of VM i at the discrete-time k, b_k is a design parameter, and $\omega_{i,k}$ is a random variable that takes values according to both external factors that impact u and Flock migrations prior to time k. For all $k \in \mathbb{Z}$, the values of ω are determined by $\omega_{i,k} = h(u_{i,k}, Y_{i,k})$ and $Y_{i,k} = \int_k^{k+1} f(m_i(t))dt$, where $m_i(t)$ is a continuous-time Markov process of generator $G^{u_{i,k}}$ and with values in a finite set $\mathcal{M}, f : \mathcal{M} \leftarrow \mathbb{R}^K$ is an arbitrary mapping, and $h : \mathbb{R}^L \times \mathbb{R}^K \leftarrow \mathbb{R}^L$ is a bounded continuous Lipschitz function in u and is uniformly distributed in Y. The Markov process $m_i(t)$ is irreducible and ergodic where $m_i(k) = \lim_{t \to k, t<k} m_i(t)$. We previously assumed that $u_{i,k}$ is bounded, which is true given the bounded latencies, VM demands, and EC capacities. We also can enforce such bounded values of u by projecting $u_{i,k}$ to a finite subset of $\mathbb{R}^L$. Finally, we assume that b_k is positive and decreasing in k, such that b_k is constant as $k \to \infty$, so that $\sum_k b_k = \infty$ and $\sum_k b_k^2 < \infty$.

By adopting the discrete time factor into the values of $u_{i,k}$, we transfer Flock into a class of stochastic approximation algorithms with controlled continuous-time Markov noise [15]. As b_k is a decreasing step size, the speed of variations in $u_{i,k}$ decreases and vanishes as k increases. This is equivalent to modeling $m_i(t)$ as a Markov process with a fixed generator (e.g. when τ values are frozen), and converges to an ergodic behavior. Hence, as k increases, Flock uses averaged values of u_i and represents a stable dynamical system (see [15] for formal proofs of the convergence of stochastic approximation algorithms). We call the modified migration protocol, controlled-Flock and is

given as:

Controlled-Flock: Autonomous VM migration protocol.

Initialization: Each VM $i \in V$ runs at a cloud $x \in A$.

Ensure: A Nash equilibrium outcome σ.

During round k, do in parallel $\forall i \in V$

Greedy migration process imitating cohesion in flocking:

i solicits its current strategy A_i from x.

i randomly selects a target cloud $y \in A_i$.

if $u_{i,k}(y)f(w_y+u_{i,k}(y)) \leq \eta u_{i,k}(x)f(w_x-u_{i,k}(x))$ **then**

  $x \xrightarrow{i} y$.

  $x = y$

end if

Update $u_{i,k+1}(x) = u_{i,k}(x) + b_k f(w_x)$

In the above algorithm, b_k is a decreasing step size and is left as a design parameter. For example $b_k = 1/k$ satisfies the decreasing condition of b_k and that $\sum_k b_k = \infty$ and $\sum_k a_k^2 < \infty$. If b_k is constant, we would obtain weak convergence only. The function $f(w_x)$ serves as the random variable $\omega_{i,k}$ for a VM i. In such case $f(w_x) \equiv h(u_{i,k}, Y_{i,k})$, where $Y_{i,k} = \int_k^{k+1} \sum_{j:x_j=x, j\neq i} u_{j,t}(x)dt$ and $u_{j,t}(x)$ is the continuous-time value of u_j. The Markov process $m_i(t)$ in this case represents the current outcome σ^k and that $f(m_i(t)) = \sum_{j:x_j=x, j\neq i} u_{j,t}(x)$ is an arbitrary mapping that is bounded and continuous Lipschitz function. We can easily verify that the properties of stochastic approximation algorithms with controlled continuous-time Markov noise as described earlier are satisfied and convergence of Controlled-Flock is ensured given the various dynamics. Fig. 6 shows the convergence of Controlled-Flockunder network dynamics that are faster than Flock decisions. Controlled-Flockmaitains the same convergence and PoA properties of Flock. However, the actual number of migrations required to converge to a nash equilibrium is greater. This is expected as Controlled-Flockdo not reach steady state until b_k approaches a constant value as $k \to \infty$, such that rapid variations in measurements become indifferent to migration decisions.

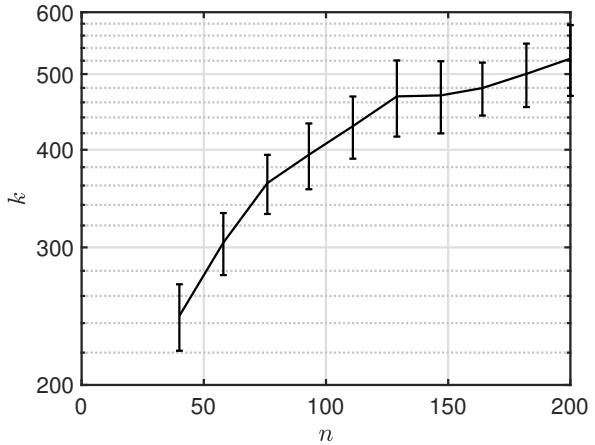

Fig. 6: Convergence of Controlled-Flock: $m = 37$, $\tau \sim$ Uniform$(10, 100)$, $d \sim$ Uniform$(1, 10)$, $a = 9$, $\gamma \sim$ Uniform$(50, 100)$, and $\eta = 0.9$. The shown values are at ± 0.02 error with 95% confidence interval.

C. Migration cost

Migration of a VM can be too costly to perform very frequently [16]. It can also degrade performance by introducing migration noise [17]. Depending on the implementation details of a VM and the workload it serves, migration can involve transferring a large volume of data between ECs, which introduces a significant network overhead. For some cloud services, frequent migration can cause intolerable services interruption, as the "down-time" due to a migration can range from few milliseconds to seconds [18]. Accounting for the migration cost in Flock can be desired for several applications. We incorporate the migration cost in Flock utility values by two methods.

First, we include the migration cost as an external dynamics in the VMs' utilities and use Controlled-Flock to minimize the utility, which also includes the average migration cost. Let $g_i(k) \in \{0, 1\}$ indicate whether i migrated at time k or not. Also let $R_i(k+1) = \beta_i R_i(k) + (1-\beta_i)g_i(k)$ denote the average forgetting value of the migrations of VM i, where $0 \leq \beta_i \leq 1$ is a VM specific parameter that reflects the impact of frequent migrations on service disruptions. The migration cost, C_i, is an increasing function of $R_i(k)$, $C_i(R_i(k))$. To incorporate the migration cost, we redefine u_i for the target cloud in Flock as

$$u_i(y) = \sum_{j\in V} d_{ij} l'(y, x_j) / \sum_{j\in V} d_{ij},$$

where $l'(y, x_j) = l(y, x_j) + C_i(R_i(k)) + C_j(R_j(k))$ and $u_i(x)$ (i.e. utility for current cloud) is memorized. This is equivalent to penalizing the latencies values measured with an additional dynamic that accounts for the estimated average number of migrations. As R_i is a function in prior migration, the C_i value increases as i migrates more frequently and vanishes over time as i pauses migrations. This tweak perceives the numerical properties of both l and u, hence maintains the convergence and PoA results of Controlled-Flock.

Although the first approach minimizes the average migration cost as $k \to \infty$, it requires exchanging the estimated migration cost between each $i, j \in E$ to maintain the reciprocity condition of l. This rapid message exchange introduces a significant communication overhead. Moreover, incorporating the average migration cost in u_i only has a long-term benefit on the system as k grows and it can causes undesirable short-term service disruption for some applications. For such interruption-sensitive applications, it may be beneficial to delay the migration decisions using the η parameter instead of incorporating the migration cost into the utility function. In this second approach each VM evaluates its own η_i accounting for the estimated number of migration as: $\eta_i(k) = \frac{1}{\exp(R(k))}$. As the VM, i, performs less migrations over time, $\eta_i(k) \to 1$ and i migrates for any slight improvement in its utility. Whereas if i performs frequent migrations, $\eta_i(k) \to 0.36$, where i only migrates if the migration decision brings a significant improvement to u_i. The drawback of this approach can be seen in Fig. 2, where we sacrifice the tightness of the PoA as $\eta < 1$.

Migration cost of an individual VM can also be reduced in practice if the VM workload does not require persistent state maintenance. For example, if a VM is running device's cloning function as in [19], it can be sufficient to only migrate device's

meta-data (few kilobytes) and use it to start a new cloning VM at the target cloud. We apply this trick in developing a message brokering system (see [20]) to minimize the messaging latency for IoT devices. For advanced workload types, systems such as SonicMigration and adaptive pre-paging ([17], [21]) can be used to minimize the VM migration overhead by examining the memory pages in a fine-grain manner and transfer only memory pages that are necessary for an application.

V. Conclusion

We propose Flock; a simple autonomous VM migration protocol. Flock considers the peer-to-peer interaction of VMs in heterogeneous edge and conventional cloud platforms. We show that Flock converges to a Nash equilibrium with $(1+\epsilon)$ PoA. Flock minimizes the average latency of VMs as a generic goal with diverse use cases and can be easily redesigned to serve other purposes such as load-balancing and energy efficiency.

VI. Acknowledgment

This work was made possible by NPRP grant # NPRP 5-319-2-121 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors. The authors would like to thank Bassem Khalfi and the anonymous reviewers for their comments to make this paper more readable.

References

[1] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 1–14.

[2] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: at the leading edge of mobile-cloud convergence," in *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*. IEEE, 2014, pp. 1–9.

[3] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205–228, 2015.

[4] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *Cloud Computing, IEEE Transactions on*, vol. PP, no. 99, 2015.

[5] L. Chen, H. Shen, and K. Sapra, "Distributed autonomous virtual resource management in datacenters using finite-markov decision process," in *Proceedings of the ACM Symposium on Cloud Computing*. ACM, 2014, pp. 1–13.

[6] R. Tron and R. Vidal, "Distributed computer vision algorithms," *Signal Processing Magazine, IEEE*, vol. 28, no. 3, pp. 32–45, 2011.

[7] C. Ding, B. Song, A. Morye, J. Farrell, A. K. Roy-Chowdhury *et al.*, "Collaborative sensing in a distributed ptz camera network," *Image Processing, IEEE Transactions on*, vol. 21, no. 7, pp. 3282–3295, 2012.

[8] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*. ACM, 2013, pp. 15–20.

[9] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Cloud of things for sensing as a service: Sensing resource discovery and virtualization," in *Global Telecommunications Conference (GLOBECOM 2015), 2015 IEEE*. IEEE, 2015.

[10] S. Abdelwahab, B. Hamdaoui, and M. Guizani, "Bird-vne: Backtrack-avoidance virtual network embedding in polynomial time," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 4983–4989.

[11] K. Bhawalkar, M. Gairing, and T. Roughgarden, "Weighted congestion games: The price of anarchy, universal worst-case examples, and tightness," *ACM Transactions on Economics and Computation*, vol. 2, no. 4, p. 14, 2014.

[12] N. Matloff, "Introduction to discrete-event simulation and the simpy language," *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August*, vol. 2, p. 2009, 2008.

[13] S. Chien and A. Sinclair, "Convergence to approximate nash equilibria in congestion games," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 169–178.

[14] C. Kubrusly and J. Gravier, "Stochastic approximation algorithms and applications," in *1973 IEEE Conference on Decision and Control including the 12th Symposium on Adaptive Processes*, no. 12, 1973, pp. 763–766.

[15] V. S. Borkar, "Stochastic approximation with controlled markovnoise," *Systems & control letters*, vol. 55, no. 2, pp. 139–145, 2006.

[16] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *IEEE International Conference on Cloud Computing*. Springer, 2009, pp. 254–265.

[17] A. Koto, H. Yamada, K. Ohmura, and K. Kono, "Towards unobtrusive vm live migration for cloud computing platforms," in *Proceedings of the Asia-Pacific Workshop on Systems*. ACM, 2012, p. 7.

[18] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.

[19] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 301–314.

[20] S. Abdelwahab and B. Hamdaoui, "Fogmq: A message broker system for enabling distributed, internet-scale iot applications over heterogeneous cloud platforms," *arXiv preprint arXiv:1610.00620*, 2016.

[21] M. R. Hines and K. Gopalan, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*. ACM, 2009, pp. 51–60.