AIMD scheduling and resource allocation in distributed computing systems

Eleftherios Vlahakis, Nikolaos Athanasopoulos, Seán McLoone

Abstract— **We consider the problem of simultaneous scheduling and resource allocation of an incoming flow of requests to a set of computing units. By representing each computing unit as a node, we model the overall system as a multi-queue scheme. Inspired by congestion control approaches in communication networks, we propose an AIMD-like (additive increase multiplicative decrease) admission control policy that is stable irrespective of the total number of nodes and AIMD parameters. The admission policy allows us to establish an event-driven discrete model, triggered by a locally identifiable enabling condition. Subsequently, we propose a decentralized resource allocation strategy via a simple nonlinear state feedback controller, guaranteeing global convergence to a bounded set in finite time. Last, we reveal the connection of these properties with Quality of Service specifications, by calculating local queuing time via a simple formula consistent with Little's Law.**

I. Introduction

Distributed computing is a new paradigm emerging to address the growing demand for extensive, real-time computations at the *edge* as a result of the growing number of end-users (e.g., smart devices, sensors) connected to the edge of the Internet. Although this emerging technology opens new opportunities for more sophisticated applications (see, e.g., [1]–[3]), it presents several research challenges, especially in the context of resource allocation and control of edge-servers due to factors such as the need to take account of latency constraints, limited capacity of edge-servers, and its inherent decentralized structure.

Feedback control has been a powerful mathematical tool for tackling management problems in the context of modern computer systems [4]. Given a representative dynamical model, control theory allows analytical derivation of formal guarantees and certificates. However, modelling computer systems is a formidable task by itself, thus, many works rely on application-specific models obtained via system identification methods. See for example [5]–[7]. Focusing on a more abstract modelling paradigm agnostic to each individual node specificities, we follow a queueing system modelling approach that enhances scalability, naturally, at the expense of accuracy loss. Notable works avoiding application-specific modelling can be found in [8]–[10].

The control problem considered in this paper consists of 1) the scheduling, and 2) the resource allocation of a stream of requests associated with a specific application, to a set of computing units. Representing each computing unit as a node and associating each node with a queue, we model the entire scheme as a multi-queue system. We assume that there is no interaction between nodes and computing units are independent from each other. A central node acts as an aggregation point, receiving all requests and dispatching them to individual nodes. Queues in this work are consistent with the *First Come First Served* (FCFS) selection policy.

Our approach to scheduling and resource allocation is motivated by the Additive Increase Multiplicative Decrease (AIMD) algorithm, a celebrated method in network management. The AIMD algorithm was originally introduced in [11] for tackling congestion phenomena in computer networks in a robust and decentralized manner requiring minimum interaction between nodes. Since then, it has become a fundamental building block of the Transmission Control Protocol (TCP) widely used across the Internet. An excellent and comprehensive study of the AIMD algorithm with several extensions and applications can be found in [12].

In this paper, we introduce an AIMD-inspired simple control policy for general scheduling problems. A typical AIMD model results in an event-driven discrete controller which is triggered by an event associated with capacity constraints, e.g., bandwidth constraints. Berman *et al.* in [13] and Shorten *et al.* in [14] show that such a control scheme can be formulated as a positive system, thus, stability and convergence properties can be derived from the Perron–Frobenius Theorem. A first challenge we face is that a positive system formulation is not possible in our case due to the absence of a capacity constraint in a scheduling task. Instead, we consider a queue clearance event and manage to show stability via a significant result in Linear Algebra (cf. [15], [16]) involving the eigenproblem of rank-one perturbations of symmetric matrices (Theorems 1 and 2). This formulation leads to a new admission control algorithm with AIMD structure which is stable irrespective of the AIMD tuning and the number of nodes, and inherits attractive features of the standard AIMD algorithm (e.g., fairness among nodes, tunable convergence rate) [17]. To the best of our knowledge, this paper presents the first admission control policy with AIMD dynamics for scheduling tasks.

As a result of the simplicity of the AIMD scheduling policy proposed, we subsequently formulate a resource allocation strategy defined as a decentralized globally stabilizing nonlinear feedback controller. We show that under the proposed resource allocation law, individual queues are bounded, and, further, converge in finite time to a well-

*This work is supported by the CHIST-ERA 2018 project DRUID-NET "Edge Computing Resource Allocation for Dynamic Networks" https://druidnet.netmode.ntua.gr/

Authors are with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Northern Ireland, UK. E-mail addresses: {e.vlahakis, n.athanasopoulos, s.mcloone}@qub.ac.uk

defined interval which is invariant [18]. This effectively permits analysis of Quality of Service (QoS) metrics, such as queueing time. Overall, scheduling and resource allocation lie in the same control loop leading to a simple decentralized system which is stable, scalable, and locally configurable.

Unlike standard stochastic methods, see, e.g., [19], [20], we follow a deterministic approach to workload modelling. This choice simplifies the simultaneous scheduling and resource allocation problem, and most importantly, leads to deterministic performance certificates.

The remainder of the paper is organized as follows. Useful definitions and assumptions are given in Section II. The main results of the paper, namely, the AIMD scheduling strategy, the resource allocation control, and the calculation of queueing time are presented in Sections III, IV, and V, respectively. In Section VI we highlight our results via an illustrative numerical example. Finally, Section VII discusses our main results and future research directions.

II. Definitions and Basic Assumptions

A. Single-queue system

We define a *request* as an individual demand for computing resources provided by a computing node. A *computing node* is defined as the physical (or virtual) computing environment, consisting of hardware, software, and network resources, whereby a request is executed. A *queue* is defined as the waiting mechanism whereby requests arriving at a node are temporarily put on hold until being selected for service. Here, we consider queues consistent with the *First Come First Served* (FCFS) selection principle.

A *queueing system* [21], [22] is defined as the dynamic relationship developed between requests entering and exiting a computing device in the presence of a queue. The number of queued requests at time t, is defined as the difference between arrivals and departures in interval $[0,\, t]$. Typically, arrival and departure processes are considered as stochastic and described by appropriate probability distributions. A comprehensive overview of stochastic queueing systems can be found in [21]. Here, to highlight the admission and resource allocation control strategies proposed in the paper, we focus on deterministic models and assume that: 1) the arrival rate, denoted by $\lambda(t)$, is constant, and; 2) requests arriving at a queueing system are identical in terms of computing resources required to serve them, such as, CPU time, memory, and disk space.

B. Event-driven discretization and event generator

An event generator is introduced as the mechanism indicating time instants at which a well-defined triggering condition $\mathscr{C}_\varepsilon$ is satisfied. Condition satisfaction can be written as $\mathscr{C}_\varepsilon(t_k) = \texttt{true}$, where t_k denotes the time instant at which the kth event occurs (is generated). The evolution of time events can be modelled as $t_{k+1} = t_k + T(k)$, where $T(k)$ may be time-varying.

In the following, we construct an aperiodic model (with respect to time), the derivation of which is the result of two crucial design strategies, namely: 1) the introduction of a *batch* queue into the system, and; 2) the adoption of an AIMD admission control policy. This strategic choice is now exemplified via a simple tandem queueing system.

C. Tandem queueing system with AIMD dynamics

We consider the two-queue system (also termed tandem queueing system) shown in Fig. 1, where $\lambda(t)$ is a piecewise differentiable function representing workload, $\delta(t)$ is the number of queued requests waiting at queue Q_1 to be dispatched to queue Q_2 at an admission rate $u(t)$, while $w(t)$ and $\gamma(t)$ represent queued requests and service rate, respectively. Using this notation, the continuous-time dynamics of

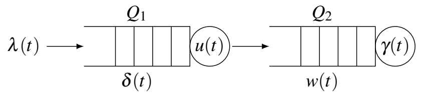

Fig. 1: A tandem queueing system.

the two-queue system can be written in a compact form as

$$\begin{bmatrix} \dot{\delta}(t) \\ \dot{w}(t) \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} \lambda(t) \\ u(t) \\ \gamma(t) \end{bmatrix}. \tag{1}$$

Before proceeding with the discretization of the model, we define a triggering condition that enables the generation of events stipulating the update of control variables $(u(t),\, \gamma(t))$. Let $u(t)$ be an admission control policy such that

$$\delta(t_k) = 0, \tag{2}$$

i.e., all the requests that have arrived at queue Q_1 by time t_k have been admitted to queue Q_2. Hence, in this regard, Q_1 instantaneously becomes empty at t_k. To ensure that condition (2) can always be satisfied at a finite time for a constant $\lambda(t) > 0$, we design the admission rate $u(t)$ as an AIMD controller as follows. Let

$$u(t_k^-) = \lim_{\substack{t \to t_k \\ t < t_k}} u(t), \;\; u(t_k^+) = \lim_{\substack{t \to t_k \\ t > t_k}} u(t), \tag{3}$$

where t_k is the instant at which the kth event occurs. Since $\delta(t_k^+) = 0$, we let

$$u(t_k^+) = \beta u(t_k^-), \tag{4}$$

where $0 < \beta < 1$ is called the *backoff parameter*. Intuitively, this means that by the time queue Q_1 becomes empty, the admission control $u(t)$ instantaneously shrinks to a fraction of $u(t^-)$ according to (4). This is called the *Multiplicative Decrease* (MD) phase. Since $u(t_k^+) < u(t_k^-)$, queue Q_1 starts growing. Right after the MD phase, the admission rate $u(t)$ increases in a ramp fashion as

$$u(t) = \beta u(t_k^-) + \alpha(t - t_k),\; t \geq t_k, \tag{5}$$

where the slope of the ramp $\alpha > 0$ is called the *growth rate*. Since $u(t)$ is strictly increasing in t, there exists a time $t_{k+1} > t_k$ such that $\delta(t_{k+1}) = 0$. We call $T(k) = t_{k+1} - t_k$ the *inter-event period*, and the interval $(t_k^+,\, t_{k+1}^-)$ the *Additive Increase* (AI) phase.

Denoting time instants t_{k+1}, t_k by integers $k+1$, k, with $k \geq 0$, an event-driven discrete model of the system shown in Fig. 1, associated with the AIMD admission policy described above, is derived as

$$\begin{bmatrix} w(k+1) \\ u(k+1) \end{bmatrix} = \begin{bmatrix} w(k) + (\beta u(k) + \frac{\alpha}{2} T(k) - \gamma(k)) T(k) \\ \beta u(k) + \alpha T(k) \end{bmatrix}, \quad (6)$$

where $u(k)$ is an AIMD controller with triggering condition $\delta(k+1) = \delta(k) = 0$. Next, we generalize the AIMD admission control approach to a system with multiple queues, and examine the properties of the AIMD algorithm and its effect on the entire system dynamics.

III. AIMD Admission Control

We consider a set of n computing nodes represented by a multi-queue system as depicted in Fig. 2. Each node is modelled by a queue combined with a (physical or virtual) computing environment. We assume that a constant workload λ enters the system via a batch queue, which is independent of the computing nodes. The workload is manifested as a flow of requests that are dispatched to n computing units (nodes) according to an admission control policy $u_i(t)$, $i = 1, \ldots, n$. We denote the number of queued requests that have not yet been admitted at time t by $\delta(t)$, and the number of admitted requests waiting to be selected for service by node-i at time t by $w_i(t)$. The service rate of node-i is denoted by $\gamma_i(t)$, $i = 1, \ldots, n$. Next, we adopt the AIMD admission control policy and the event-driven formulation described in Sections II-B and II-C.

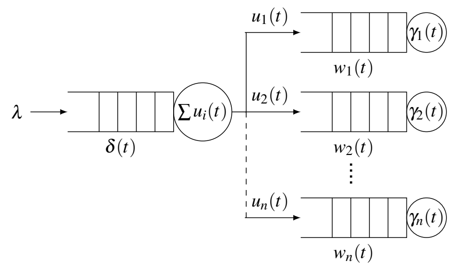

Fig. 2: A multi-queue system with AIMD admission control policy.

The number of queued, thus unadmitted, requests at the beginning of the $(k+1)$th event is given by

$$\delta(k+1) = \delta(k) + \lambda T(k) - \int_{t_k}^{t_{k+1}} \sum u_i(t) \mathbf{d}t. \quad (7)$$

We recall that at each event k, $k+1$, ..., we have

$$\delta(k) = \delta(k+1) = \cdots = 0. \quad (8)$$

We now show that our admission control yields an exact formula for the inter-event period $T(k)$, thus, permitting a closed form of the aggregate admission control system. During the AI phase, the ith admission rate ramps up as follows,

$$u_i(t) = \beta_i u_i(t_k) + \alpha_i (t - t_k), \; i = 1, \ldots, n, \quad (9)$$

which is a continuous-time controller for $t \in [t_k, \; t_{k+1})$. Based on condition (8), the event-driven dynamics of the ith admission controller is written as:

$$u_i(k+1) = \beta_i u_i(k) + \alpha_i T(k), \; i = 1, \ldots, n. \quad (10)$$

In view of the triggering condition (8) and using (10) in (7), we get $\lambda T(k) = \sum_{i=1}^{n} (2\beta_i u_i(k) + \alpha_i T(k)) \frac{T(k)}{2}$[1] or $\lambda = \sum_{i=1}^{n} (\beta_i u_i(k) + \frac{\alpha_i}{2} T(k))$, from which, the inter-event period is defined as

$$T(k) = \frac{\lambda - \sum_{i=1}^{n} \beta_i u_i(k)}{\sum_{i=1}^{n} \frac{\alpha_i}{2}}. \quad (11)$$

From (11), we may write that $u_i(k+1) = \beta_i u_i(k) + \alpha_i \frac{\lambda - \sum_{i=1}^{n} \beta_i u_i(k)}{\sum_{i=1}^{n} \frac{\alpha_i}{2}}$. Defining $u(k) = (u_1(k), \ldots, u_n(k))$, $\bar{\alpha} = \frac{1}{\sum_{j=1}^{n} \alpha_j} (\alpha_1, \ldots, \alpha_n)$, $B = \text{diag}(\beta_1, \ldots, \beta_n)$, $\beta = (\beta_1, \ldots, \beta_n)$ the aggregate admission control system can be expressed as

$$u(k+1) = \Phi u(k) + 2\bar{\alpha}\lambda, \quad (12)$$

where $\Phi = B - 2\bar{\alpha}\beta'$, with $\mathbf{1}'\bar{\alpha} = 1$, $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^n$. Next, we show that system (12) is stable, thus, $u(k)$ converges to a unique equilibrium point u^*. We first present the following result, which appears in several works in the context of Linear Algebra, see, e.g., [16, Theorem 1], [15, Section 5].

Theorem 1 *Let $C = D + \rho z z'$, where $D \in \mathbb{R}^{n \times n}$ is diagonal, $\rho \in \mathbb{R}$, and $z \in \mathbb{R}^n$. Let $d_1 \leq d_2 \leq \ldots \leq d_n$ be the eigenvalues of D, and $c_1 \leq c_2 \leq \ldots \leq c_n$ be the eigenvalues of C. Then,*

i.) $d_1 \leq c_1 \leq d_2 \leq c_2 \leq \ldots \leq d_n \leq c_n$ *if* $\rho > 0$,
ii.) $c_1 \leq d_1 \leq c_2 \leq d_2 \leq \ldots \leq c_n \leq d_n$ *if* $\rho < 0$.

We are in a position to state the first main result, namely, the stability of the AIMD scheduling policy.

Theorem 2 *Let vectors $\bar{\alpha} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_n)$, $\beta = (\beta_1, \ldots, \beta_n)$, where $0 \leq \bar{\alpha}_i \leq 1$, $0 < \beta_i < 1$, $\forall i = 1, \ldots, n$, and $\mathbf{1}'\bar{\alpha} = 1$, with $\mathbf{1} = (1, \ldots, 1)$. Let also $B = \text{diag}(\beta_1, \ldots, \beta_n)$. Then,*

$$\Phi = B - 2\bar{\alpha}\beta', \quad (13)$$

is a Schur matrix.

Proof: Matrix Φ can also be written as

$$\Phi = (I - 2A)B, \quad (14)$$

where $A = \bar{\alpha}\mathbf{1}'$ is a rank-one matrix with $\sigma(A) = \{\mathbf{1}'\bar{\alpha}, 0, \ldots, 0\}$, and $\mathbf{1}'\bar{\alpha} = 1$ by definition. In the sequel, we denote by $\sigma(\Phi) = \{\phi_1, \ldots, \phi_n\}$ the spectrum of Φ. Clearly, $\sigma(B) = \{\beta_1, \ldots, \beta_n\}$. Also, it is easy to show that $\sigma(I - 2A) = \{-1, 1, \ldots, 1\}$. We can also write that $\det(\Phi) = \phi_1 \phi_2 \ldots \phi_n$, and $\det(\Phi) = \det(B) \det(I - 2A)$. Thus,

$$\phi_1 \phi_2 \cdots \phi_n = -\beta_1 \beta_2 \cdots \beta_n. \quad (15)$$

Let now $\hat{A} = \text{diag}(\bar{\alpha}_1, \ldots, \bar{\alpha}_n)$, and $\hat{\Phi} = B^{\frac{1}{2}} \hat{A}^{-\frac{1}{2}} \Phi \hat{A}^{\frac{1}{2}} B^{-\frac{1}{2}}$. Clearly, matrices Φ and $\hat{\Phi}$ are similar, and therefore have identical eigenvalues. Note also that $\hat{\Phi}$ can be written as

[1]the integral on the right side of (7) is the sum of areas of n trapezoids due to the AIMD dynamics of u_i, i = 1,..., n.

$\hat{\Phi} = B - 2zz'$, which is clearly a symmetric matrix, where $z = (\sqrt{\bar{\alpha}_1\beta_1}, \ldots, \sqrt{\bar{\alpha}_n\beta_n})$. Without loss of generality, let $b_1 \leq \ldots \leq b_n$, and $\phi_1 \leq \ldots \leq \phi_n$. Then, from Theorem 1, and since all elements of z are nonzero, we may write that

$$\phi_1 \leq \beta_1 \leq \phi_2 \leq \beta_2 \leq \ldots \leq \phi_n \leq \beta_n. \tag{16}$$

From (15) and (16), we can conclude that $0 < \phi_2, \phi_3, \ldots, \phi_n < 1$, and ϕ_1 is a negative real number. From (16), we have that

$$\phi_2\phi_3 \cdots \phi_n \geq \beta_1\beta_2 \cdots \beta_{n-1}. \tag{17}$$

However, due to (15), (17) implies that $|\phi_1| \leq \beta_n$, i.e., $-\beta_n \leq \phi_1 < 0$. Thus, all the eigenvalues of $\hat{\Phi}$ (hence Φ), strictly lie inside the unit circle (specifically on the real axis between -1 and 1). This proves the theorem. ■

The main deductions that follow from the analysis presented in this section are as follows:

1) AIMD parameters can be individually selected at the node level. Thus, under the reasonable assumption that each node is aware of when an event occurs, i.e., when $\delta(t) = 0$, the proposed admission control policy (10) is fully decentralised.
2) In view of Theorem 2, the aggregate system (12) is stable regardless of the choice of AIMD parameters.
3) The ith AIMD admission rate converges to $u_i^* = \frac{\alpha_i}{1-\beta_i}T^*$, where $T^* = \sum_{j=1}^{n}(\frac{\alpha_j}{2}\frac{1+\beta_j}{1-\beta_j})^{-1}\lambda$.

IV. Resource Allocation Control

Resource allocation in queueing systems pertains to strategies ensuring that computing nodes provide application requests with adequate resources so that queueing time is bounded as more requests are added to the system. Stability, minimizing queueing times and computing costs, as well as maximizing system throughput are essential objectives. Here, we focus on stability as a fundamental qualitative property, in the absence of which, any other desirable objective of a queueing scheme may be impossible to attain.

We follow a bottom-up approach for designing a decentralized resource allocation control strategy as follows. Let (α_i, β_i) be the AIMD parameters, and $\gamma_i(k)$ denote the service rate at the kth event, respectively, associated with node-i. Recall that between the kth and $(k+1)$th events,

$$u_i(\tau) = \beta_i u_i(k) + \alpha_i\tau, \ \tau \in [0, T(k)], \tag{18}$$

is the rate at which requests are admitted to node-i, while

$$w_i(\tau) = w_i(k) + \beta_i u_i(k)\tau + \frac{\alpha_i}{2}\tau^2 - \gamma_i(k)\tau, \ \tau \in [0, T(k)], \tag{19}$$

is the number of queued requests waiting in node-i. We define by

$$y_i^k(\tau) = w_i(\tau) + \gamma_i(k)\tau, \ \text{and} \ z_i^k(\tau) = \gamma_i(k)\tau, \tag{20}$$

the total number of requests that have been admitted, and the number of requests that can be served at most, respectively, in $(t_k, \tau]$ with $\tau \in (t_k, T(k))$. Let also $\hat{\gamma}_i(k)$ be the slope of a line segment starting from the origin and being tangent to parabola $y_i^k(\tau)$ (see $O\Delta$ in Fig. 3). By letting $\gamma_i(k) = \hat{\gamma}_i(k)$, thus selecting z_i^k as the line segment tangent to $y_i^k(\tau)$ at point $t_z^k \in [0, T(k)]$, as shown in Fig. 3, we effectively guarantee that the maximum number of requests that can be served between events never exceeds the actual number of admitted requests, thus, avoiding node under-utilization, or respectively, over-provisioning of resources. In Theorem 3 below, we also show that this resource allocation choice is stabilizing. Note also that if $\gamma_i(k) > \hat{\gamma}_i(k)$ (see red dashed line in Fig. 3) there is always a nonzero interval that the queue of node-i remains empty, i.e., resources may be unreasonably over-provisioned. Similarly, by letting $0 < \gamma_i(k) < \hat{\gamma}_i(k)$ (see blue dashed line in Fig. 3) there is no an obvious stability guarantee indicating that the ith queue remains bounded.

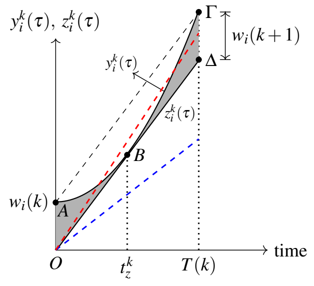

Fig. 3: Arrivals and departures in node-i.

We now show how to obtain a closed formula for $\hat{\gamma}_i(k)$. We first find the intersection point B, as shown in Fig. 3, where

$$y_i^k(t_z^k) = z_i^k(t_z^k), \tag{21}$$

$$\frac{\mathrm{d}y_i^k}{\mathrm{d}t_z^k} = \frac{\mathrm{d}z_i^k}{\mathrm{d}t_z^k}. \tag{22}$$

From (22), we get $t_z^k = \frac{\hat{\gamma}_i(k) - \beta_i u_i(k)}{\alpha_i}$, while, using the latter in (21), after a few calculations, we have

$$\hat{\gamma}_i(k) = \beta_i u_i(k) + \sqrt{2\alpha_i w_i(k)}, \tag{23}$$

which is an event-driven, nonlinear state-feedback controller. Finally, using (23), we have $t_z^k = \sqrt{\frac{2w_i(k)}{\alpha_i}}$. We are now in a position to state the proposed resource allocation strategy along with its stability properties.

Theorem 3 *Let*

$$w_i(k+1) = w_i(k) + (\beta_i u_i(k) + \frac{\alpha_i}{2}T(k) - \gamma_i(k))T(k), \tag{24}$$

with $w_i(0) \geq 0$, be the queue dynamics of node-i, where (α_i, β_i) are AIMD parameters, and $T(k)$ is the inter-event period. Consider the resource allocation policy

$$\gamma_i(k) = \beta_i u_i(k) + \sqrt{2\alpha_i w_i(k)}. \tag{25}$$

Then, the following hold.

i.) *System (24)-(25) is nonnegative for all $w_i(k) \geq 0$.*
ii.) *The set $\mathscr{W}_i(k) = [0, \frac{a_i}{2}T(k)^2]$ is invariant with respect to system (24)-(25).*

iii.) *For $w_{i,0} \notin \mathscr{W}_i(k)$, there is an integer $k^\star > 0$ such that $w_i(k^\star) \in \mathscr{W}_i(k)$.*

Proof: i.) In the proof, we denote $\frac{\mathrm{d}\phi}{\mathrm{d}x}$ by $\phi'(x)$. Substituting (25) in (24), we write

$$w_i(k+1) = w_i(k) + \frac{\alpha_i}{2}T(k)^2 - \sqrt{2\alpha_i w_i(k)}T(k), \qquad (26)$$

and we show that $f(w_i(k)) = w_i(k+1)$ is convex in $w_i(k)$. Indeed, $f(w_i(k))$ is convex with respect to $w_i(k)$ since it is a sum of the affine function $w_i(k) + \frac{\alpha_i}{2}T(k)^2$ and the convex function $-\sqrt{2\alpha_i w_i(k)}T(k)$. Note also that $f(w_i(k))$ is convex for all $T(k) \geq 0$. Since, $f(w_i(k))$ is continuously differentiable and convex for $w_i(k) \geq 0$, the unique minimizer is attained by setting $f'(w_i(k)) = 0$, which results in $1 - \frac{\sqrt{2\alpha_i}T(k)}{2\sqrt{w_i(k)}} = 0$, or $w_i^*(k) = \frac{\alpha_i T(k)^2}{2}$. Taking into account that $f(w_i^*(k)) = 0$, it holds that $f(w_i(k)) \geq 0 \ \forall\, w_i(k) \geq 0$, $T(k) \geq 0$.

ii.) The condition $w_i(k+1) \leq w_i(k)$ holds if $w_i(k) + \frac{\alpha_i}{2}T(k)^2 - \sqrt{2\alpha_i w_i(k)}T(k) \leq w_i(k)$, i.e., $w_i(k) \geq \frac{\alpha_i}{8}T(k)^2$.

Let $\hat{\mathscr{W}}_i(k) = \left\{w \in \mathbb{R} : w \geq \frac{\alpha_i}{8}T(k)^2\right\}$. Since $\hat{\mathscr{W}}_i(k) \cap \mathscr{W}_i(k) = [\frac{\alpha_i}{8}T(k)^2,\ \frac{\alpha_i}{2}T(k)^2]$, we need only to verify $w_i(k+1) \leq w_i(k)\ \forall w_i(k) \in [0,\ \frac{\alpha_i}{8}T(k)^2)$. Since $f(w_i(k))$ is convex with minimum at $\frac{\alpha_i}{2}T(k)^2$ it follows that $f(0) \geq f(w_i(k))$ for any $w_i(k) \in [0,\ \frac{\alpha_i}{2}T(k)^2]$. Since $f(0) = \frac{\alpha_i}{2}T(k)^2$, it holds that $0 \leq f(w_i(k)) \leq \frac{\alpha_i}{2}T(k)^2$ for all $w_i(k) \in \mathscr{W}_i(k)$. This proves part *ii.)*.

iii.) Consider function $g(w_i(k)) = w_i(k) - f(w_i(k))$. Then, $g'(w_i(k)) = 1 - f'(w_i(k)) = \frac{\sqrt{2\alpha_i}T(k)}{2\sqrt{w_i(k)}} > 0$ since $T(k) > 0$ for all $k \geq 0$. Moreover, $g(\frac{\alpha_i}{2}T(k)^2) = \frac{\alpha_i}{2}T(k)^2$. Thus, $\forall k > 0$, and $\forall w_i(k) \geq \frac{\alpha_i}{2}T(k)^2$, we have $g(w_i(k)) \geq \frac{\alpha_i}{2}T(k)^2$, or $f(w_i(k)) \leq w_i(k) - \frac{\alpha_i}{2}T(k)^2$. We now claim that for any $w_i(0) \geq \frac{\alpha_i}{2}T(0)^2$, $\exists\ k^*$ such that $w_i(k^*) \leq \frac{\alpha_i}{2}T(k^*)^2$. Indeed, $w_i(k^*) \leq w_i(0) - \sum_{j=0}^{k^*-1}\frac{\alpha_i}{2}T(j)^2$. To enforce the claim, we have $w_i(0) - \sum_{j=0}^{k^*-1}\frac{\alpha_i}{2}T(j)^2 \leq \frac{\alpha_i}{2}T(k^*)^2$, i.e., $w_i(0) - \sum_{j=0}^{k^*}\frac{\alpha_i}{2}T(j)^2 \leq 0$, which is satisfied if $w_i(0) - (\frac{\alpha_i}{2}\min_{j=0,\ldots,k^*}T(j)^2)(k^*+1) \leq 0$, or, $k^* \geq \left\lceil \frac{2w_i(0)}{\alpha_i \min_{j=0,\ldots,k^*}T(j)^2} - 1 \right\rceil$, which can always be found. ■

It is worth highlighting some appealing characteristics of the proposed resource allocation scheme:

1) System (24) under the resource allocation policy (25) is globally attracted to $\mathscr{W}_i(k) = [0,\ \frac{\alpha_i}{2}T(k)^2]$ in finite time.
2) Policy (25) is decentralised as only local information is required. Moreover, it is scalable with respect to the number of computing nodes.
3) Stability properties of (24)-(25) are independent of the particular tuning of AIMD parameters α_i, β_i.

V. Queuing Time Calculation

In the following, we use a standard definition of queueing time consistent with *Little's Law*, see, e.g., [21, Chapter 2]. We define the total queueing time associated with requests dispatched to node-i as

$$Q_i(k) = Q_{\delta_i}(k) + Q_{w_i}(k), \qquad (27)$$

where $Q_{\delta_i}(k)$ corresponds to queueing time in the batch queue, while $Q_{w_i}(k)$ corresponds to queueing time in node-i. The average admission rate associated with node-i is defined as $u_i^{\mathbf{av}}(k) = \frac{1}{T(k)}\int_{t_k}^{t_{k+1}} u_i(t)\mathbf{d}t$, where $T(k) = t_{k+1} - t_k$, and $u_i(t)$ is given by (9). Solving the integral above yields

$$u_i^{\mathbf{av}}(k) = \beta_i u_i(k) + \frac{\alpha_i}{2}T(k). \qquad (28)$$

Substituting $T(k)$ in (28) from (11), we get

$$\sum_{i=1}^{n} u_i^{\mathbf{av}}(k) = \lambda,\ \forall\, k \geq 0. \qquad (29)$$

In view of (29), we can write that $u_i^{\mathbf{av}}(k)T(k)$ corresponds to the fraction of the total arrivals at the batch queue, namely, $\sum_{i=1}^{n} u_i^{\mathbf{av}}(k)T(k) = \lambda T(k)$ associated with node-i.

We define $\delta_i(\tau) = u_i^{\mathbf{av}}(k)\tau - \beta_i u_i(k)\tau - \frac{\alpha_i}{2}\tau^2$, with $0 \leq \tau \leq T(k)$, as the number of queued requests waiting in the batch queue before being dispatched to node-i. Note that $\sum_{i=1}^{n}\delta_i(\tau) = \delta(\tau)$, and $\delta_i(t_k) = 0$ for all $i = 1,\ldots,n$. We calculate $Q_{\delta_i}(k) = \frac{\int_0^{T(k)}\delta_i(\tau)\mathbf{d}\tau}{u_i^{\mathbf{av}}(k)T(k)}$, and see that integral $\int_0^{T(k)}\delta_i(\tau)\mathbf{d}\tau$ is the unshaded area enclosed by parabola $AB\Gamma$ and the line segment $A\Gamma$, in Fig. 3. Similarly, we calculate $Q_{w_i}(k) = \frac{\int_0^{T(k)} w_i(\tau)\mathbf{d}\tau}{\beta_i u_i(k)T(k) + \frac{1}{2}\alpha_i T(k)^2}$, where integral $\int_0^{T(k)} w_i(\tau)\mathbf{d}\tau$ is the shaded area in Fig. 3, and $\beta_i u_i(k)T(k) + \frac{1}{2}\alpha_i T(k)^2 = u_i^{\mathbf{av}}(k)T(k)$ due to (28). Adding the two aforementioned areas and dividing by $u_i^{\mathbf{av}}(k)T(k)$ clearly yields the queueing time associated with node-i. In other words, $Q_i(k)$ is equal to the area enclosed by the trapezium $A\Gamma\Delta O$ divided by $u_i^{\mathbf{av}}(k)T(k)$, i.e.,

$$Q_i(k) = \frac{w_i(k) + w_i(k+1)}{2u_i^{\mathbf{av}}(k)}. \qquad (30)$$

Remark 1 *The total queueing time of node i can be defined by means of local information without any information pertinent to the batch queue. Also, letting $w_i^{\mathbf{av}}(k) = (w_i(k) + w_i(k+1))/2$ be the average number of queued requests between two successive events,* (30) *becomes $Q_i(k) = \frac{w_i^{\mathbf{av}}(k)}{u_i^{\mathbf{av}}(k)}$, which is clearly consistent with Little's Law.*

VI. Numerical Example

We consider requests entering a system of four computing nodes at a constant rate λ. Each node admits requests according to (10), and alters its service rate via (25). Simulation results are presented in Fig. 4-6 for the setup parameters shown in Table I.

λ	α_i	β_i	$u_i(0)$	i
100	$5i$	0.5	$(i-1)5$	$\{1,2,3,4\}$
$\mathscr{W}_1(k)$	$\mathscr{W}_2(k)$	$\mathscr{W}_3(k)$	$\mathscr{W}_4(k)$	$w_i(0)$
[0, 4.44]	[0, 8.88]	[0, 13.33]	[0, 17.77]	$(2i-1)7$

TABLE I: Simulation parameters / Invariant sets

As can be seen from Fig. 4(left), the inter-event period converges to $T^* = 1.33$ [sec]. Viewing Fig. 5(left), admission rates also converge to $u_i^* = \frac{\alpha_i}{1-\beta_i}T^*$, $i = 1,\ldots,4$, verifying the

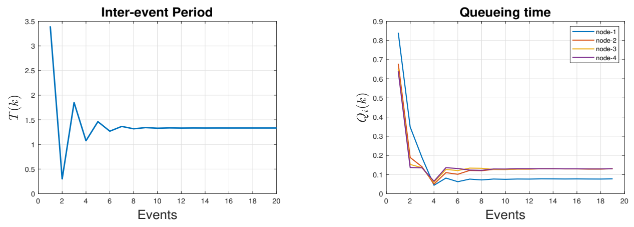

Fig. 4: Left: Inter-event period. Right: Queueing time.

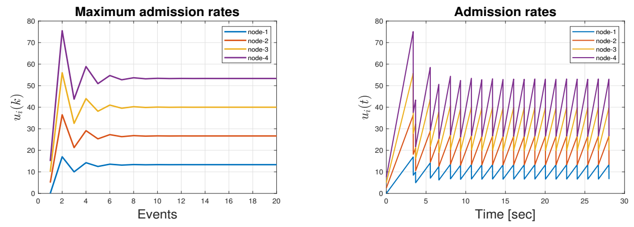

Fig. 5: Left: Admission rates at events. Right: Admission rates at time t.

validity of Theorem 2. Fig. 5(right) illustrates typical AIMD behaviour with convergence occurring after approximately 10 events. Faster convergence may be attained if α_i, $i = 1,\ldots,4$, are selected more aggressively. Service rates are depicted in Fig. 6(left). It is evident that $\gamma_i(k)$ are strongly related to $u_i^{\mathbf{av}}(k)$. Queue profiles are shown in Fig. 6(right), where it is seen that queues are bounded highlighting the stability properties of Theorem 3. Invariant sets $\mathscr{W}_i(k)$ for $T(k) = T^*$ are given in Table I. Overall, stable operation is obvious and guaranteed for all computing nodes regardless of the tuning of individual AIMD parameters. We refer interested readers to [23] for further simulation scenarios with arbitrary numbers of nodes and random arrival processes.

VII. Conclusion

We study the problem of simultaneous scheduling and resource allocation of requests entering a system of computing nodes. Inspired by the well-established AIMD algorithm, we present a new admission control policy for general scheduling problems. We provide stability guarantees, independent of the overall system dimension and AIMD tuning. Following a bottom-up approach, we then propose a globally stabilizing resource allocation strategy defined as an event-driven decentralized nonlinear feedback controller. This effectively guarantees that individual queues are bounded converging in finite time to an interval. Finally, we associate these properties with Quality of Service specifications. Our method is simple, scalable, and locally configurable. Two important challenges, namely, non deterministic workload and the presence of resource constraints will be addressed in future work.

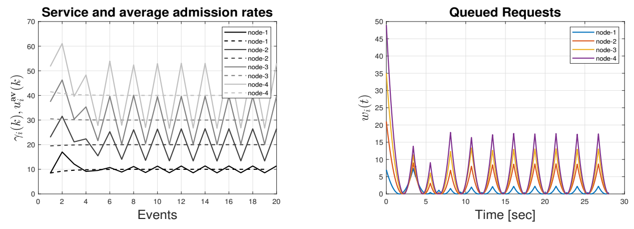

Fig. 6: Left: Service rates (solid lines) and average admission rates (dashed lines). Right: Queued Requests.

References

[1] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A Survey of Research on Cloud Robotics and Automation," *IEEE Trans. on Automat. Science and Engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[2] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE IoT Journal*, vol. 5, pp. 450–465, 2018.

[4] C. Karamanolis, M. Karlsson, and X. Zhu, "Designing controllable computer systems," in *Proc. of 10th Workshop on Hot Topics in Operat. Sys.* Berkeley: USENIX Association, 2005, pp. 9–15.

[5] Z. Wang, X. Zhu, and S. Singhal, "Utilization and SLO-based control for dynamic sizing of resource partitions," in *16th IFIP/IEEE Ambient Networks international conference on Distributed Systems: Operations and Management*. Springer Verlag, 2005, pp. 133–144.

[6] D. Dechouniotis, N. Leontiou, N. Athanasopoulos, A. Christakidis, and S. Denazis, "A control-theoretic approach towards joint admission control and resource allocation of cloud computing services," *Intern. Journal of Network Management*, vol. 25, no. 3, pp. 159–180, 2015.

[7] M. Avgeris, D. Dechouniotis, N. Athanasopoulos, and S. Papavassiliou, "Adaptive resource allocation for computation offloading: A control-theoretic approach," *ACM Transactions on Internet Technology*, vol. 19, no. 2, pp. 1–20, 2019.

[8] M. Maggio, H. Hoffmann, M. D. Santambrogio, A. Agarwal, and A. Leva, "Controlling software applications via resource allocation within the Heartbeats framework," in *Proceedings of the IEEE Conference on Decision and Control*, 2010, pp. 3736–3741.

[9] E. Kalyvianaki, T. Charalambous, and S. Hand, "Adaptive resource provisioning for virtualized servers using kalman filters," *ACM Transactions on Autonom. and Adaptive Syst.*, vol. 9, no. 2, pp. 1–35, 2014.

[10] E. Makridis, K. Deliparaschos, E. Kalyvianaki, A. Zolotas, and T. Charalambous, "Robust Dynamic CPU Resource Provisioning in Virtualized Servers," *IEEE Transactions on Services Computing*, 2020.

[11] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, 1989.

[12] M. Corless, C. King, R. Shorten, and F. Wirth, *AIMD Dynamics and Distributed Resource Allocation*. Society for Industrial and Applied Mathematics, 2016.

[13] A. Berman, R. Shorten, and D. Leith, "Positive matrices associated with synchronised communication networks," *Linear Algebra and Its Applications*, vol. 393, no. 1-3, pp. 47–54, 2004.

[14] R. N. Shorten, D. J. Leith, J. Foy, and R. Kilduff, "Analysis and design of AIMD congestion control algorithms in communication networks," *Automatica*, vol. 41, no. 4, pp. 725–730, 2005.

[15] G. Golub, "Some Modified Matrix Eigenvalue Problems," *SIAM Review*, vol. 15, no. 2, pp. 318–334, 1973.

[16] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, no. 1, pp. 31–48, 1978.

[17] R. Shorten, F. Wirth, and D. Leith, "A positive systems model of TCP-like congestion control: Asymptotic results," *IEEE/ACM Transactions on Networking*, vol. 14, no. 3, pp. 616–629, 2006.

[18] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*, ser. Systems & Control: Foundations & Applications. Birkhäuser, 2015.

[19] S. Maguluri and R. Srikant, "Scheduling jobs with unknown duration in clouds," in *Proc. - IEEE INFOCOM*, 2013, pp. 1887–1895.

[20] S. Maguluri, R. Srikant, and L. Ying, "Heavy traffic optimal resource allocation algorithms for cloud computing clusters," *Performance Evaluation*, vol. 81, pp. 20–39, 2014.

[21] L. Kleinrock, *QUEUEING SYSTEMS, Volume I: Theory*. Wiley, 1975.

[22] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer US, 2008.

[23] [Online]. Available: https://github.com/lefterisvl83/cdc21