2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Progress Rate Control for Computer Applications

Alberto Leva, Silvano Seva and Alessandro Vittorio Papadopoulos

Abstract— **Self-adaptive software applications often include some form of progress rate control. Various frameworks were proposed to measure progress and provision resources to govern it, hence - in control terms - for sensors and actuators. The same is not true for control laws, however. In this paper we address this part of the overall problem, proposing a standard control structure that can be easily configured and tuned to match a variety of progress control needs. We completely analyse the simplest case, namely a single application under fixed rate control, and spend a few words on extensions to multiple application and event-based realisation. Simulation examples are reported to support the proposal.**

I. Introduction

Control theory is gaining importance as a means to design and assess components of computing systems. This applies to several domains, from real-time and embedded systems [1]–[6], to software engineering [7]–[11], from resource allocation problems [12]–[14], to cloud computing [15]–[22]. Often, such systems are called "self-adaptive " or "autonomic computing" systems, depending on the community.

A control-theoretic design approach is particularly effective when reliable models for the dynamics of the process to be controlled are available. When dealing with computing systems this is often not the case, as in general they are not ruled by physical laws [23] at a macroscopic level, so that identification techniques are frequently brought into play [7], [10], [23].

However, there exist cases in which dynamic models of computing system components can be obtained from first-principle-like laws, based for example on the queueing theory [19], [21], or on intuitive considerations. A notable one is encountered when controlling the progress of a quantity towards a given goal. This case takes a number of forms. It appears in real-time scheduling, where a budget of time must be allocated to a specific process to execute a given work so as to not exceed a deadline [2]–[5]. It plays a role in synchronization schemes, where discrepancies in the frequencies of the crystal oscillators aboard nodes in a network require action to control the difference among their clocks [6]. In these and analogous situations, the process to be controlled can be though of as a single integrator: in the first example, the amount of executed work accumulates over time and it to reach a specific threshold within a deadline; in the second example, the synchronization error accumulates over time if no control action is taken.

In the case just mentioned, the role of control ultimately amounts to governing the progress rate of the state of an integrator, the input of which is the manipulated variable subjected to additive and/or multiplicative disturbances—a problem that can be named "(application) progress control". Despite its extreme simplicity, such a model of the controlled system does allow to tackle real-life problems, as shown e.g. in [6], [24]. Hence, studying the properties of this particular control problem has a notable practical relevance. The goal of this paper is to address the problem in a general fashion.

II. Related work and motivation

There is a vast number of literature works on progress control. Since a full review is impossible here, we just mention a few examples to motivate the presented research.

The first one is the Heartbeat framework [25] that "instruments" the application to emit a "heartbeat" (via a system call) whenever a meaningful set of operations have been executed. For example, a video encoder can emit a heartbeat for every encoded frame. This is a progress measure, that can be used e.g. to dynamically allot resources so as to maintain a prescribed frame rate despite different videos result in very different encoding workloads [26].

Another example is [27], where the authors consider heterogeneous multi-processing for runtime self-adaptive multithreaded applications, and propose a framework for monitoring and dynamically adapting the application behaviour to enhance its performance and power consumption with respect to user-specified goals. The presented approach, however, is strongly dependent on space exploration, which limits scalability. Similarly, in [28], the authors propose a proactive approach for self-adaptive software systems, that seeks the optimal strategy on a finite set of actions that needs to be explored.

Then, in [29], a performance management system is presented that acts at the operating system scheduler level and uses progress information to achieve "performance-aware" fairness, designing an Observe-Design-Act control loop. The resulting control law is a heuristic, hence not providing guarantees on convergence towards the goals, as the proposal is an extension of the best-effort Completely Fair Scheduler.

Even on the sole basis of the examples just reported, one can observe that the way progress/performance is measured is quite well assessed, and the same is true for the way to act on resources, meaning that for both these aspects the various proposals are quite similar to one another. However, the same is not true at all as for the way decisions are taken.

A. Leva is with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy. alberto.leva@polimi.it

S. Seva is a student at the Dipartimento di Elettronica, Informazione e Bioingegneria. silvano.seva@mail.polimi.it

A.V. Papadopoulos is with the School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden alessandro.papadopoulos@mdh.se

There is thus room for introducing a modelling- and control-based design approach to standardise the control part of the application progress loop, which is the main point of our proposal.

III. The proposed control scheme

A. Dynamic model

The accomplishment $a_c(t)$ of the application task, i.e., the amount of processed data, can be described by

$$a_c(t) = \int_0^t K_r \eta_c(\tau) r(\tau) d\tau. \tag{1}$$

where $r(t)$ is the *amount of resource* allotted to the application, e.g., the share of cores, $K_r > 0$ is the *resource gain*, i.e., how much a unit of resource contributes to the accomplishment of the application task, and $\eta_c(t)$ is the *resource efficiency*, that is a number limited as $0 < \eta_{\min} \leq \eta_c(t) \leq 1$, accounting for possible non-complete availability of the resource, e.g., when part of the allotted share is unexpectedly taken by other tasks.

Since in a computing system decisions can be made only at discrete time instants, every h time units, one can discretize (1) as

$$\begin{aligned} a_c(kh) =: a(k) =& a(k-1) + \int_{(k-1)h}^{kh} K_r \eta_c(\tau) r(\tau) d\tau \\ =& a(k-1) + K_r h \eta(k) r(k-1). \end{aligned} \tag{2}$$

where

$$\eta(k) := \frac{1}{h} \int_{(k-1)h}^{kh} \eta_c(\tau) d\tau. \tag{3}$$

Despite its extreme simplicity, this model already proved to be suitable for a control-oriented dynamic description of a task pool [3], [24]. In progress control, the desired behaviour of a is most frequently a ramp-like set point. Hence, we adopt as the controller a discrete-time PI, that in state-space form reads

$$\begin{cases} x_C(k) = x_C(k-1) + b_R e(k-1) \\ r(k) = x_C(k) + d_R e(k) \end{cases} \tag{4}$$

where $e(k) := a^\circ(k) - a(k)$ is the error. The closed-loop dynamics is thus described by the LPV system

$$x(k) = A(k) x(k-1) + b(k) a^\circ(k-1), \tag{5}$$

with state $x(k) = \begin{bmatrix} a(k) & x_C(k) \end{bmatrix}^\top$, and

$$A(k) = \begin{bmatrix} 1 - K_r h d_R \eta(k) & K_r h \eta(k) \\ -b_R & 1 \end{bmatrix}, \; b(k) = \begin{bmatrix} K_r h d_R \eta(k) \\ b_R \end{bmatrix}. \tag{6}$$

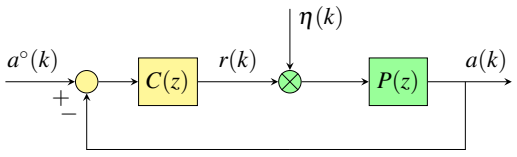

Fig. 1: Discrete-time progress control scheme.

This system is shown as block diagram in Figure 1, where $C(z)$ is the transfer function of (4), and the time-varying efficiency is modelled as a (bounded) multiplicative disturbance.

B. Stability analysis

Proposition 3.1 (Theorem 1 in [30]): Two linear time-invariant systems Σ_{A_1} and Σ_{A_2} with dynamic matrices A_1, $A_2 \in \mathbb{R}^{2\times 2}$, both constant and Schur matrices, have a Common Quadratic Lyapunov Function if and only if the matrix pencils $H(\alpha, A_1, A_2)$ and $H(\alpha, A_1, -A_2)$ are Schur, with

$$H(\alpha, A_1, A_2) = (0.5I - G(\alpha, A_1, A_2))^{-1}(0.5I + G(\alpha, A_1, A_2)),$$
$$G(\alpha, A_1, A_2) = \alpha P + (1-\alpha) Q,$$

where $\alpha \in [0,1]$, and $P := 0.5I - (I + A_1)^{-1}$, and $Q := 0.5I - (I + A_2)^{-1}$.

Requiring $H(\alpha, A_1, A_2)$ and $H(\alpha, A_1, -A_2)$ to be Schur, is equivalent to require $G(\alpha, A_1, A_2)$ and $G(\alpha, A_1, -A_2)$ to be Hurwitz [30, Section 4]. In particular, letting $P = [p_{ij}]$, and $Q = [q_{ij}]$, and

$$r_1 = p_{11} q_{22} + q_{11} p_{22} - p_{12} q_{21} - q_{12} p_{21} \tag{7}$$
$$r_2 = \det(Q) \tag{8}$$
$$r_3 = r_1 - 2 r_2 \tag{9}$$
$$r_4 = \det(P) + r_2 - r_1. \tag{10}$$

the following holds.

Proposition 3.2 (Section 4 in [30]): $G(\alpha, A_1, A_2)$ is Hurwitz if and only if one of the following conditions is satisfied

(i) $r_4 \leq 0$, or
(ii) $r_4 > 0$, $-r_3/2r_4 \notin [0,1]$, or
(iii) $r_4 > 0$, $-r_3/2r_4 \in [0,1]$, and $r_2 - r_3^2/4r_4 > 0$

Applying this to our problem, we can now state our results.

Lemma 3.3: For any value of $0 < \eta_{\min} \leq \eta(k) \leq 1$, K_r, and h, there exists a couple of values (b_R, d_R) that make the origin of the system (5) a globally asymptotically stable equilibrium for the closed loop system.

Proof: The closed-loop dynamic matrix of the system (5) can be written as

$$A(k) = \alpha(k) A_1 + (1 - \alpha(k)) A_2, \quad 0 \leq \alpha(k) \leq 1 \forall k \tag{11}$$

with

$$A_1 = \begin{bmatrix} 1 - K_r h d_R \eta_{\min} & K_r h \eta_{\min} \\ -b_R & 1 \end{bmatrix}, \tag{12}$$

$$A_2 = \begin{bmatrix} 1 - K_r h d_R & K_r h \\ -b_R & 1 \end{bmatrix} \tag{13}$$

For convenience, we also introduce the quantity

$$\beta = \frac{2}{K_r h}. \tag{14}$$

Both A_1 and A_2 are Schur if and only if

$$b_R > 0 \tag{15}$$
$$b_R < d_R \tag{16}$$
$$b_R > 2 d_R - 2\beta. \tag{17}$$

To illustrate our results, we hereinafter refer to Figure 2, the yellow solid triangles denote the region $R_1 := (15)\bigcap(16)\bigcap(17)$, where A_1 and A_2 are Schur.

If we compute r_4 as per (10) for the pencil $G(\alpha,A_1,A_2)$, we obtain $r_4 = 0$; thus for Proposition 3.2, $G(\alpha,A_1,A_2)$ is Hurwitz. Therefore, in the following we focus on the pencil $G(\alpha,A_1,-A_2)$.

Computing r_4 for the pencil $G(\alpha,A_1,-A_2)$ gives

$$r_4 = \frac{2(d_R - b_R)(1+\eta_{\min})K_r h - 4}{K_r^2 h^2 \eta_{\min} b_R \left(2d_R - b_R - \frac{4}{K_r h \eta_{\min}}\right)}. \tag{18}$$

The value of r_4 is negative or equal to zero if the numerator and denominator in (18) have different sign, i.e., when

$$\begin{cases} b_R \le d_R - \dfrac{\beta}{1+\eta_{\min}} \\ b_R > 2d_R - \dfrac{2\beta}{\eta_{\min}} \end{cases} \tag{19}$$

$$\begin{cases} b_R \ge d_R - \dfrac{\beta}{1+\eta_{\min}} \\ b_R < 2d_R - \dfrac{2\beta}{\eta_{\min}} \end{cases} \tag{20}$$

Intersecting (19)–(20) with R_1, we obtain

$$R_2 := \begin{cases} b_R \le d_R - \dfrac{\beta}{1+\eta_{\min}} \\ b_R > 2d_R - 2\beta \\ b_R > 0 \end{cases} \tag{21}$$

that is indicated in Figure 2 by the red cross-hatched triangles. The intersection with the axes of R_1 and R_2 have been also reported. It is clear that $\eta_{\min} \to 0$, R_2 collapses to the point $(\beta,0)$, and for any value of $\eta_{\min} > 0$, there are always values of b_R and d_R that make the origin of system (5) globally asymptotically stable.

For completeness, we proceed to study the second condition of 3.2, with the $-r_3/2r_4$ for $G(\alpha,A_1,-A_2)$. This quantity takes the form

$$-\frac{r_3}{2r_4} = \frac{(4d_R - 3b_R)(1+\eta_{\min})K_r h + d_R(b_R - d_R)\eta_{\min}K_r^2 h^2 - 8}{4(d_R - b_R)(1+\eta_{\min})K_r h - 8}. \tag{22}$$

Since we are considering the case $r_4 > 0$ and we are interested in the region R_1, thanks to (19), we can say that the denominator of (22) is always negative. The numerator of (22) is positive for

$$b_R > \widetilde{b}_R(d_R) := \frac{8 - 4d_R(1+\eta_{\min})K_r h + d_R^2 K_r^2 \eta_{\min} h^2}{d_R \eta_{\min} K_r^2 h^2 - 3(1+\eta_{\min})K_r h}, \tag{23}$$

hence making (22) negative in the region R_3 denoted by the blue diagonal hatch. The intercept of R_3 with the d_R axis is for $\overline{d_R} = 2\beta/(1+\eta_{\min}+\sqrt{1+\eta_{\min}^2})$, and for $\eta_{\min} \to 0$, it will collapse to the point $(\beta,0)$.

The intersection of R_3 with $R_1 \setminus R_2$ provides another set of values of b_R and d_R that makes the origin of system (5) globally asymptotically stable. ■

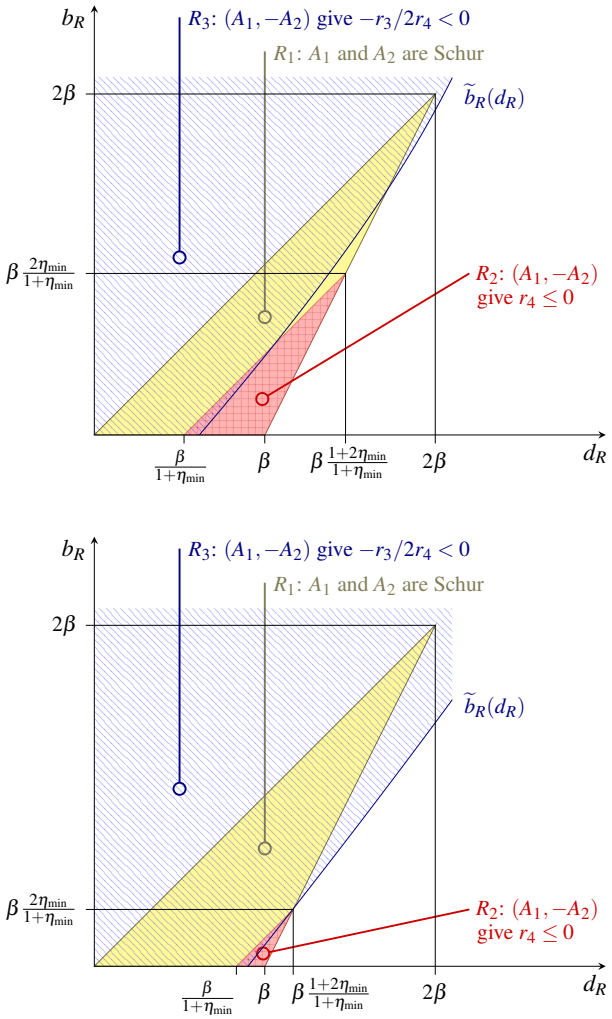

Fig. 2: Stability region in the (d_R,b_R) plane for $\eta_{\min} = 0.9$ (top) and $\eta_{\min} = 0.2$ (bottom).

Theorem 3.4: If $\eta_{\min} \to 0$, then the origin is globally asymptotically stable for system (5) if and only if A_1 and A_2 are Schur.

Proof: Thanks to Lemma 3.3 we identified the regions R_2 and $R_3 \bigcap R_1$ of the values d_R and b_R that make the origin of system (5) globally asymptotically stable. The union of R_2 and $R_3 \bigcap R_1$ may or may not cover the entire R_1 depending on $\eta_{\min}$, as shown by the two situations in Figure 2. However, for $\eta_{\min} \to 0$, R_2 tends to collapse into the $(\beta,0)$ point, while $R_3 \bigcap R_1$ tends to cover the entire R_1. This concludes the proof. ■

In practice, if the efficiency of the controlled application falls below a certain threshold, the hosting machine is so heavily loaded that the only sensible choice is to shut off the application on that machine and run it elsewhere. However, nothing prevents to suppose that the minimum possible efficiency is (arbitrarily close to) zero, which guarantees stability in the whole (open) region R_1.

C. Tuning

We refer to the block diagram of Figure 1, where for constant η we have

$$L(z) = P(z)C(z) = \frac{K_r h \eta}{z-1}\left(\frac{b_R}{z-1} + d_R\right). \tag{24}$$

For convenience, as specifications are most frequently given as desired completion times, we now re-interpret the control system in the continuous time. By inverse forward Euler, and for a constant η, the loop transfer function takes the form

$$L(s) = P(z)C(z)|_{z=1+hs} = \frac{K_r \eta b_R}{h}\frac{1 + s\frac{h d_R}{b_R}}{s^2} \tag{25}$$

whose frequency response magnitude is depicted in Figure 3.

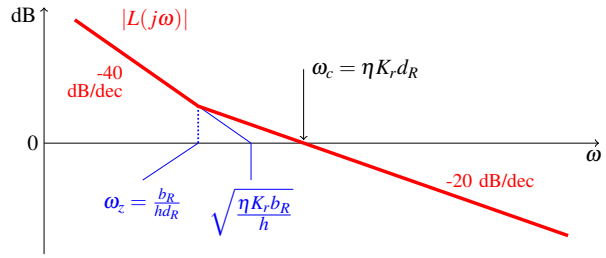

Fig. 3: Magnitude of the loop frequency response as seen in the continuous time.

The cutoff frequency ω_c and the phase margin φ_m can be expressed respectively as

$$\omega_c = \eta K_r d_R \tag{26}$$

$$\varphi_m = \arctan\left(\frac{\omega_c}{\omega_z}\right) = \arctan\left(\frac{2\eta}{\beta}\frac{d_R^2}{b_R}\right). \tag{27}$$

The behaviour of φ_m for some values of η is shown in Figure 4, where the axes are normalised with respect to β, and the stability region is indicated on the base plane.

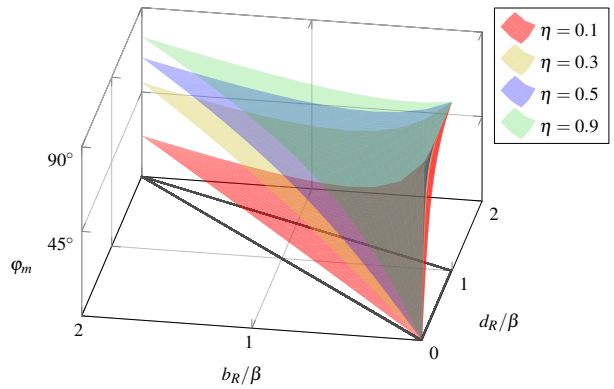

Fig. 4: Phase margin as a function of (d_R, b_R) in the stability region, for different constant values of η

For the purpose of tuning $C(z)$, we take as the first design parameter a value of $\eta_{\min}$ below which we decide that the application is just to be shut off and relocated. In Section III-B we showed that this is stability-safe. Given this, and having measured K_r with the initial calibration experiment described in Section III-D, we choose d_R, b_R and h by solving the system

$$\begin{cases} b_R = \frac{K_r h \eta_{\min}}{\tan \overline{\varphi}_m} d_R^2 \\ d_R = \frac{2}{K_r h} \\ d_R = \frac{\overline{\omega}_c}{K_r \eta_{\min}} \end{cases} \tag{28}$$

that takes as further specifications a desired cutoff frequency $\overline{\omega}_c$ and phase margin $\overline{\varphi}_m$, and gives

$$d_R = \frac{\overline{\omega}_c}{K_r \eta_{\min}}, \quad b_R = \frac{2\overline{\omega}_c}{K_r \tan \overline{\varphi}_m}, \quad h = \frac{2\eta_{\min}}{\overline{\omega}_c}. \tag{29}$$

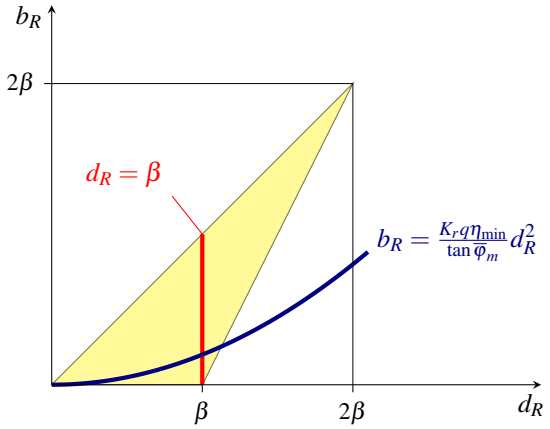

Fig. 5: Interpretation of the PI tuning as per (28).

System (28) can be interpreted graphically as in Figure 5. The thick vertical line corresponds to $d_R = \beta$, as per the second equation in (28). The obtained (d_R, b_R) couple is the intersection of this line with the parabola given by the first equation in (28). There can be different tuning policies, but this one allows to contextually select the PI parameters and h so as to stay within the stability region limiting the effect of the variations of η on the stability degree—which conversely would require to stay near to the $(\beta, 0)$ point, see Figure 4. This degree is evaluated on a hypothetical system with *constant* η just as a sensitivity clue, of course. Stability under time-varying η was already ensured, the point here is not to have excessively oscillatory behaviours of the controlled variable. Figure 5 allows also to define feasibility limits, because the two lines must cross within the stability region, but we omit this matter for space reasons. As a final remark on the contextual selection of h, we conversely observe that the sampling frequency turns out to be

$$\omega_s = \frac{2\pi}{h} = \frac{\pi}{\eta_{\min}}\omega_c, \tag{30}$$

hence choices of $\eta_{\min}$ that are sensible from the operational standpoint – say around 0.2 – are safe also for the significance of the continuous-time interpretation of the loop. In fact, in the absence of clues on $\eta_{\min}$, one could equivalently

set the ω_s/ω_c ratio, as shutting off the application at higher values of the so computed equivalent efficiency is apparently harmless.

D. Implementation and operation

We here provide an extremely brief overview of the proposed control's operation. We assume that the controlled application periodically checks its accomplishment with one of the quoted frameworks, and deposits the accomplishment value in a location accessible to the controller. The controller samples this at constant rate h, under the implicit – yet extremely reasonable – assumption that h is large enough for some application progress to generally take place.

The controller first acquires the deadline τ_{DL}, the amount D of data to process and the precaution coefficient γ from the hierarchically superior levels of the application control, and determines the accomplishment set point profile as per Figure 6. Also, the controller sets ω_c so that the closed-loop settling time be less than τ_{DL}, and chooses a suitable phase margin (this can be set to say 45°–50° by default).

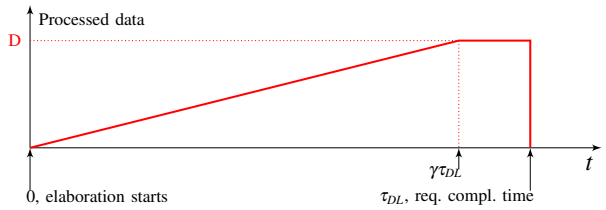

Fig. 6: Accomplishment set point profile.

Given this, for the purpose of self-calibrating, the controller allots one unit of computational resource in such a way to guarantee the application its exclusive use. This allows to measure K_r reliably, as η can be thought to be unitary, and requires short enough a time to not disturb excessively the overall system and the other applications. How this can be done is operating system-specific, and strays from the scope of this paper. Once K_r is measured and $\eta_{\min}$ set as discussed above, the tuning formulæ (29) can be applied, and control transferred to $C(z)$ until the application signals completion.

IV. Simulation examples

In this section we show two simulation examples. The proposed approach is already employed in applications [24], that incidentally motivated the analysis here reported. The typical development frameworks are complex, however, and introduce a lot of artefacts. Simulation allows to show the operation of the proposed control with better clarity. The first example is obtained by setting $\overline{\omega}_c$ to 1/15 r/s for a required completion time of 160 s with $\gamma = 0.9$, and requiring a phase margin of 40°. The accomplishment dynamics is modelled in the continuous time, i.e., as per (1). The obtained results are in Figure 7.

Qualitatively, tracking and recovery from a transient goal infeasibility (the PI has an antiwindup mechanism that we did not discuss for brevity) are good. Most important, the goal is attained with a quite long timestep. The incurred cost is

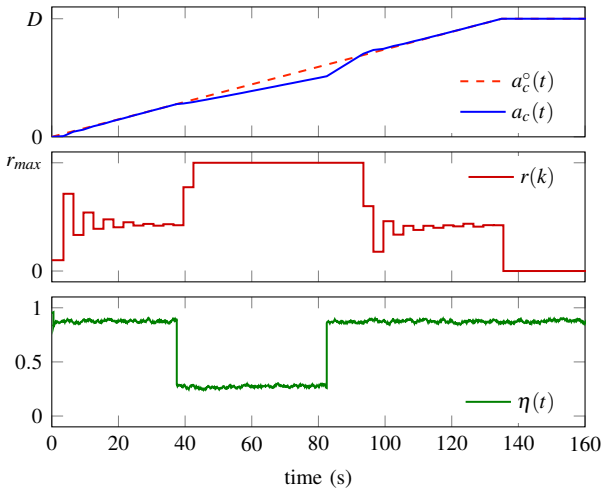

Fig. 7: Simulation example 1: a° versus a viewed in the continuous time (top), resources allotted at control steps (centre) and time-varying efficiency (bottom).

some oscillations of the control signal, but such a behaviour is well tolerable when allotting computational resources, and paid back in terms of not too frequent invocations of the controller, to the advantage of processing time left to the applications.

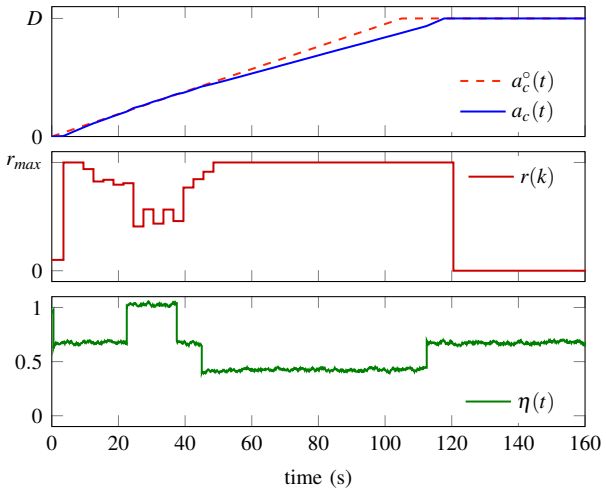

Fig. 8: Simulation example 2: figure organised in the same way as Figure 7.

The second example – shown in Figure 8 – is analogous, but the efficiency disturbance is sufficient to provoke a delay of the attainment of the final goal with respect to $\gamma\tau_{DL}$. Notice that also in this case the control signal behaves in a sane manner, despite the deadline was approached, and possibly hit if the disturbance were harsher. Trivial as it may seem, this is a major advantage of using feedback control in the place of heuristics, that can produce unpredictable results if moved off enough from their design hypotheses [3].

Also in this case, as in many others, satisfactory results

are obtained with control timesteps that are long if compared to the typical time scales of thread scheduling, but short indeed if compared to actuation schemes based on virtual machines. The proposed technique is therefore very keen to be exploited in conjunction with modern container-based resource allocators, lending itself to the advantages described in [24].

V. Extensions and conclusions

We have presented a standard control scheme for application progress control, easy to integrate into wider-scale resource and performance managers. The absence of heuristics and the self-calibration capability make this scheme useful to relief high-level decision mechanisms form the burden of managing disturbances that are local to the controlled applications. Given the wide variety of heterogeneous approaches already used for this purpose, we believe that standardisation itself in this context is a step forward.

Stability was proven and performance assessed (in simulation) for the single application case, while the multiple application one is to be studied. In this respect, plans are to exploit the fact that unless a resource constraint is hit, the controlled system is decoupled. The inherently decentralised nature of the proposed solution makes therefore intervention necessary only in the case of a contention, and the simplicity of the feedback loop should allow for simple mechanisms—possibly suboptimal, but fast enough to be applicable in situations where milliseconds are precious.

Another extension is to realise the controller in an event-based manner, allowing the application to monitor its progress, but call for a new control only in the case this deviates too much from the expected behaviour. Again, the proposed control structure is not difficult to split into local event generators, one per application, and either one allocator per application as well, or a single central one, or any combination thereof.

References

[1] L. Abeni, L. Palopoli, G. Lipari, and J. Walpole, "Analysis of a reservation-based feedback scheduler," in *23rd IEEE Real-Time Systems Symposium (RTSS)*, 2002, pp. 71–80.

[2] G. Buttazzo and L. Abeni, "Adaptive rate control through elastic scheduling," in *39th IEEE Conference on Decision and Control*, vol. 5, 2000, pp. 4883–4888.

[3] A. V. Papadopoulos, M. Maggio, A. Leva, and E. Bini, "Hard real-time guarantees in feedback-based resource reservations," *Real-Time Systems*, vol. 51, no. 3, pp. 221–246, 2015.

[4] M. Thammawichai and E. C. Kerrigan, "Feedback scheduling for energy-efficient real-time homogeneous multiprocessor systems," in *IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 1643–1648.

[5] E. Bini, "Adaptive fair scheduler: Fairness in presence of disturbances," in *24th International Conference on Real-Time Networks and Systems (RTNS)*, 2016, pp. 129–138.

[6] A. Leva *et al.*, "High-precision low-power wireless nodes' synchronization via decentralized control," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1279–1293, 2016.

[7] A. Filieri, H. Hoffmann, and M. Maggio, "Automated design of self-adaptive software with control-theoretical formal guarantees," in *36th International Conference on Software Engineering*, ser. ICSE 2014, 2014, pp. 299–310.

[8] M. Maggio *et al.*, "Self-adaptation for individual self-aware computing systems," in *Self-Aware Computing Systems*, S. Kounev, J. O. Kephart, A. Milenkoski, and X. Zhu, Eds., 2017, pp. 375–399.

[9] M. Maggio, A. V. Papadopoulos, A. Filieri, and H. Hoffmann, "Automated control of multiple software goals using multiple actuators," in *11th Joint Meeting of the european Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2017, pp. 373–384.

[10] G. A. Moreno *et al.*, "Comparing model-based predictive approaches to self-adaptation: CobRA and PLA," in *12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2017, pp. 42–53.

[11] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[12] E. Bini *et al.*, "Resource management on multicore systems: The actors approach," *IEEE Micro*, vol. 31, no. 3, pp. 72–81, 2011.

[13] A. Leva, A. V. Papadopoulos, and M. Maggio, "A general control-theoretical methodology for runtime resource allocation in computing systems," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 3487–3492.

[14] M. Maggio *et al.*, "Power optimization in embedded systems via feedback control of resource allocation," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 1, pp. 239–246, 2013.

[15] P. Padala *et al.*, "Adaptive control of virtualized resources in utility computing environments," in *2Nd ACM SIGOPS/EuroSys European Conference on Computer Systems*, 2007, pp. 289–302.

[16] M. Gaggero and L. Caviglione, "Predictive control for energy-aware consolidation in cloud datacenters," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 2, pp. 461–474, 2016.

[17] A. V. Papadopoulos and M. Maggio, "Virtual machine migration in cloud infrastructures: Problem formalization and policies proposal," in *54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 6698–6705.

[18] S. Cerf *et al.*, "Cost function based event triggered model predictive controllers application to big data cloud services," in *IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 1657–1662.

[19] J. Dürango *et al.*, "Control-theoretical load-balancing for cloud applications with brownout," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 5320–5327.

[20] A. Leva and A. V. Papadopoulos, "Modelling and control of big data frameworks," in *20th IFAC World Congress*, vol. 20, 2017.

[21] M. Kihl, A. Robertsson, M. Andersson, and B. Wittenmark, "Control-theoretic analysis of admission control mechanisms for web server systems," *World Wide Web*, vol. 11, no. 1, pp. 93–116, 2008.

[22] M. A. Kjaer, M. Kihl, and A. Robertsson, "Response-time control of a single server queue," in *46th IEEE Conference on Decision and Control*, 2007, pp. 3812–3817.

[23] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.

[24] L. Baresi, S. Guinea, A. Leva, and G. Quattrocchi, "A discrete-time feedback controller for containerized cloud applications," in *ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*, 2016, pp. 217–228.

[25] H. Hoffmann *et al.*, "Application heartbeats: A generic interface for specifying program performance and goals in autonomous computing environments," in *7th IEEE/ACM International Conference on Autonomic Computing and Communications*, 2010, pp. 79–88.

[26] M. Maggio *et al.*, "Comparison of decision making strategies for self-optimization in autonomic computing systems," *ACM Trans. on Autonomous and Adaptive Systems*, vol. 7, no. 4, 36:1–36:32, 2012.

[27] J. Yun, J. Park, and W. Baek, "Hars: A heterogeneity-aware runtime system for self-adaptive multithreaded applications," in *52nd Annual Design Automation Conference*, 2015, 107:1–107:6.

[28] G. A. Moreno, J. Cámara, D. Garlan, and B. Schmerl, "Proactive self-adaptation under uncertainty: A probabilistic model checking approach," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, 2015, pp. 1–12.

[29] F. Sironi *et al.*, "Metronome: Operating system level performance management via self-adaptive computing," in *49th Design Automation Conference*, 2012, pp. 856–865.

[30] M. Akar and K. Narendra, "On the existence of a common quadratic Lyapunov function for two stable second order LTI discrete-time systems," in *American Control Conference*, 2001, pp. 2572–2577.