Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Control-Based Resource Management for Storage of Video Streams ⋆

Alexandre Martins * **Mikael Lindberg** ** **Martina Maggio** ***
Karl-Erik Årzén ***

* *Department of Automatic Control, Lund University and Axis Communications*
** *Axis Communications*
*** *Department of Automatic Control, Lund University*

Abstract: Distributed surveillance systems typically consist of multiple cameras that need to store some fraction of their video streams at a central storage node. The disk space of this node constitutes a shared resource. In the paper the disk space allocation is formulated as a PI control problem and a new method for enforcing global resource constraints inspired by anti-windup tracking is proposed. The approach is evaluated by simulations.

Keywords: Cameras, Computer control, Constraints, PID control, Memory applications

1. INTRODUCTION

A common scenario in many control applications is the need to share some resource among a number of clients or subsystems. This happens particularly often when control is applied to computer and communication systems. In these cases a limited shared resource, e.g., communication bandwidth, CPU capacity, or memory, should be shared between a number of clients. The problem, however, does not appear only in computing systems, but can be found in other industrial sectors as well. One example is process automation, where common resources such as cooling water or steam need to be shared between several subsystems or process units.

In many cases, the amount of shared resource that should be allocated to each client is given by the output of a controller, i.e., by the control signal, which has the objective to keep some quality or performance related variable at a desired value. Hence, the overall architecture of the system consists of a number of control loops that interact with each other through the fact that the sum of all the control signals, i.e., the total amount of resources required, is limited. When there are hard or soft limitations on the total amount of resource it is also necessary to have some mechanism for prioritizing among the control loops so that the most important loop should be effected the least by the resource limitation, i.e., static priorities, or the loop that need the resources the most should be effected the least, i.e., dynamic priorities. We propose an approach that supports both options.

In this paper, the focus is storage systems for video surveillance. Video surveillance systems are increasingly prevalent in society. They are used at different levels and at different scales – e.g., cities, public places, companies, homes, etc. A typical video surveillance system comprises multiple, in some cases several hundreds or thousands, cameras, disseminated over an area and recording 24/7. Today, the video industry is mainly focused on using IP cameras, which stream videos that are compressed using the H.264 standard (Richardson, 2010), also called MPEG-4 part 10 AVC, which is currently the *de facto* standard for video encoding and decoding.

The cameras send their video streams over a network to one or several storage stations, where the video streams are monitored, e.g, by a human operator looking for abnormal events. Doing this requires the operator to have access to a sliding window of the video stream, showing, e.g., the last 15 minutes or 1 hour of the video. Storing the associated video frames requires an amount of memory that varies depending on the size of the video frames, i.e., on the dynamics of the scene. Video streams from multiple cameras typically share the available storage which then constitutes a shared resource that all cameras in the system compete for.

The allocation of storage to a video stream can be viewed as a control problem (see Fig. 1) where the measured variable is the length, or duration, of the stored video sliding window. This is compared to the desired duration, e.g., one hour, and the resulting error is fed to a controller, e.g., a PI controller, that calculates the amount of memory or disk space that the video stream may use. The frames in the video stream and the storage that they require can be viewed as a disturbance acting on the system.

We propose an approach for managing shared resources that is inspired by the tracking, or back-calculation, approach for handling anti-windup in controllers with integral actions that is commonly used in PID control. In this paper tracking is instead used to ensure that the sum of the control signals is limited.

The objective of the approach is simplicity, i.e., the approach should fit well with simple PID control schemes,

⋆ This work has been partially funded by the Wallenberg AI, Autonomous Systems and Software Program (WASP), the ELLIIT strategic research area on IT and mobile communications, and the Nordforsk university hub on Industrial IoT (HI2OT).

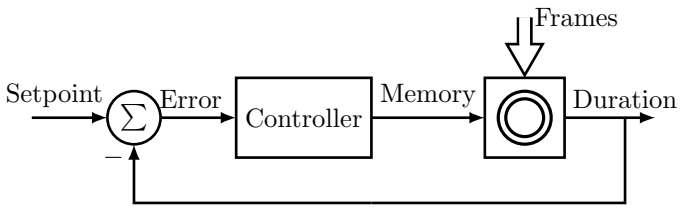

Fig. 1. Video storage feedback loop.

and to be as decentralized and asynchronous as possible. An alternative approach would be to use conventional Model-Predictive Control (MPC) (Rawlings and Mayne, 2009). However, this is more centralized and does not fit well with PID control. Also, the model of the storage used here is non-linear which would in the MPC case imply the use of non-linear MPC techniques. The price to pay for decentralization and lack of synchronization is that the global control signal constraint must be soft in nature.

The following are the contributions of the paper.

- The allocation of disk space for storing a video stream is formulated as a control problem.
- A method, that, to the best of our knowledge, is new for enforcing global constraints on the control signals for a set of controllers with integral action is proposed. The method is inspired by anti-windup tracking commonly used in PID control.
- The application of the proposed method to control of computer and communication systems where a limited resource is shared between a set of user or clients.
- The method is applied to the video storage application with promising results.

1.1 Outline of the paper

In Section 2 the model used for the video storage is described. Tracking-based anti-windup is shortly recapitulated in Section 3. The proposed general approach for handling global control signal constraints is presented in Section 4 together with a linear system example. In Section 5 the result of applying this to video storage is described. Extensions to the approach are presented in Section 6. Finally, the paper ends with suggestions for future work and conclusions in Section 7.

1.2 Related Work

Control has been applied to the problem of determining the best setting for video streaming (De Cicco et al., 2011; Cucinotta et al., 2009; Palopoli et al., 2009; Yin et al., 2015). In this case, the focus was on optimizing the video quality subject to bandwidth constraints. This problem is equivalent to meeting a certain quality of service for a real-time (Cucinotta et al., 2004) or multimedia (Palopoli et al., 2008; Cucinotta et al., 2011) application. The problem that we are facing is different. In fact, we are trying to determine what is the fraction of videos that we should store to satisfy (legal) requirements and at the same time avoid exceeding the amount of available storage. In this work we do not consider adapting the compression level of the cameras, although this would be a possibility.

The general formulation of our problem is allocating a limited resource to multiple actors. This problem has been encountered in different circumstances when controlling computing systems, e.g., in the management of a set of thread (Hellerstein et al., 2004), CPU scheduling (Leva and Maggio, 2010), or core allocation (Maggio et al., 2010). However, the solutions that were found either require domain knowledge or do not scale well. We aim at providing a general mechanism to handle the problem of partitioning the shared resource among multiple competing actors requiring the least possible amount of domain knowledge. In our case the actors are the cameras and their associated storage controllers. The objective has been an approach that is as decentralized as possible and fits well into a PID control setting.

2. STORAGE OF VIDEO STREAMS

Video storage is usually done at central locations, which could be edge storage, e.g. a computer with hard drive(s) at each geographical location or global storage, e.g. in a data center. Usually, the camera system is designed/expected to stream on average a certain amount of data per fixed duration, e.g. a Gigabit per day, week, or month. The amount of disk space allocated is then calculated with some safety margin. This storage behaves like a ring-buffer where the oldest content is deleted to allow new content to be stored. It can be deleted due to requirements (new video frames need to be stored in the system) or for legal/policy reasons (the video content should not be stored longer than a certain time). In this paper we only consider the first case: recycling for memory re-use.

The storage cost has a a large impact for companies. Hence, they try to minimize the amount of storage needed, while fulfilling the requirements in terms of duration, resolution, quality, etc. This is true especially when many video streams should be stored simultaneously.

In our work, we consider the dual problem: we are given a fixed amount of available global storage, i.e., disk space and we want to optimize its usage. We want to keep as much video as possible, satisfying given quality constraints. Our measurement variable is the stored video duration of the produced video, e.g. the amount of past video being stored in memory for each video stream. This problem can be viewed as a distributed control problem where each camera has a video recording duration set-point, e.g. camera 1 should save video for 2 days, while camera 2 should save it for 1 day. Cameras will generate video data based on their environment and configuration. They require a certain amount of disk space to be able to meet the recording duration set-point.

In a camera system, multiple cameras are competing for the same pool of storage and need then to adjust also based on the global constraint. The situation in case of two cameras is shown in Fig. 2.

The open loop performance of the ring-buffer model is illustrated in Fig. 3. The figure shows the stored video time, the amount of memory/disk space allocated to the buffer, and the frame size. We assume that initially the buffer is full. After a while the allocated storage increases by a step. This causes the stored video time to grow

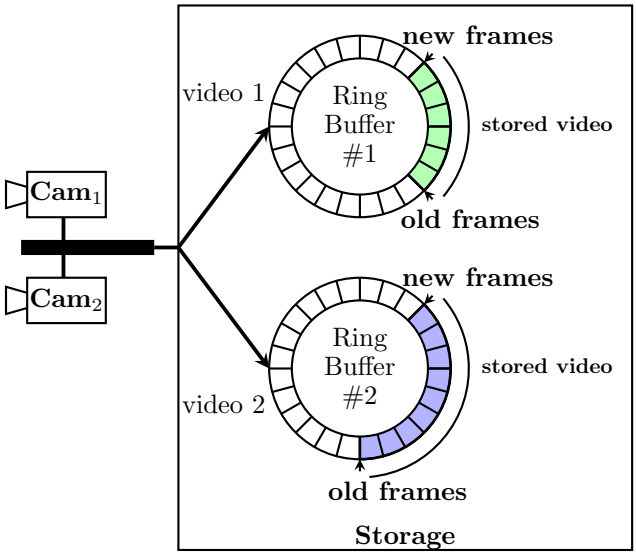

Fig. 2. Multiple video streams sharing the same disk space.

linearly as new frames are entered into the buffer until the buffer becomes full again. After a while the allocated storage is decreased, again using a step. This causes the stored video to drop instantaneously as a number of frames will be flushed from the buffer. At $t = 3000$ the average frame size is decreased. This causes the stored video time to increase linearly as more frames will fit in the buffer. Similarly when the frame size increases the stored video time will decrease linearly as there is room for fewer frames. Hence the model consists of a saturated integrator where the gain depends on the frame size and frame rate in combination with an instantaneous change when data is flushed.

The corresponding closed loop performance is shown in Fig. 4. Here a discrete-time PI controller with a sampling rate equal to the frame rate, which we assume is constant and equal to 30 fps, is used as the controller. However, also other controller types could be used as long as they contain integral action. The ring buffer is implemented as a Simulink s-function. The plot shows the stored video time including the set-point value of 900 seconds, the allocated storage, and the average frame size. At $t = 1000$ the average frame size increases and the stored video time drops. This causes the controller to increase the allocated storage until the stored video time returns to the set-point. At $t = 4000$ the frame size decreases and the stored video time consequently increases. The controller reacts by reducing the allocated storage until the stored video time is back at the set-point.

The actual size of the frames that are used in the simulations varies substantially from frame to frame. A sign of this is the rather noisy stored video time plot in Fig. 4. The reason for this is the nature of a H.264 stream. The stream consist of a sequence of *groups of pictures* (GOP). Each GOP consist of one I-frame followed by a sequence of P-frames and B-frames. I-frames are usually self-contained, i.e., they contain a full image and do not need additional information for the decoding. The P and B-frames are encoded using information contained in other frames. As a result of this the I-frames are substantially larger than the P- and B-frames. In the simulation a stream consisting of frames with the sizes shown in Fig. 5 has been used, i.e., the I-frames are 5 times as large as the other frames. This has consequences for the ring buffer storage. For example, entering a new I-frame may cause several old P- and B-frames to be removed. The overall size of all the frames depends on things such frame resolution, camera noise, scene illumination, compression level, motion level, etc, according to the model in Edpalm et al. (2018)

3. TRACKING-BASED ANTI-WINDUP

We propose a scheme for managing the allocation of shared resources, that is inspired by the tracking approach for handling anti-windup in controllers with integral action that is commonly used in PID control (Astrom and Murray, 2008).

A controller with integral action together with an actuator that becomes saturated can cause problems. If the control error is so large that the integrator causes the control signal to saturate the actuator, then the feedback loop will be broken. The reason for this is that the actuator will remain saturated also if the plant output changes. The integrator may then integrate up to a very large value. When the error finally becomes small again, the integral value may be so large that it takes a considerable amount of time until the integral assumes a normal value again. This effect is known as integrator (or reset) windup and typically causes over and undershoots in the output response.

The tracking or back-calculation approach to anti-windup is based on the addition of an extra feedback loop inside the controller. The feedback is generated by adding a simple saturation model of the actuator, and forming an error signal e_s as the difference between the estimated, possibly saturated, actuator output u and the output v that the controller would like to send out. This error is then fed back to the integrator through a gain $1/T_t$. When the actuator is not saturated the error e_s is zero and the controller is not affected by the extra feedback. When the actuator is saturated the extra feedback loop will try to force e_s to zero, by modifying the value of the integrator. This means that the integrator is reset (or back-calculated), so that the controller output is at the saturation limit. The reset is done with a time constant T_t, also known as the tracking time constant, rather than instantaneously. One reason for not this is to avoid that the integrator is reset erroneously, for example due to measurement noise.

A PI controller with tracking-based anti-windup is shown in Fig. 6. The corresponding code for the continuous-time PI controller in Equation 1

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int^t e(s)\,ds\right), \tag{1}$$

when the I-part is discretized using forward approximation is given by the following very commonly used pseudo-code adopted from Wittenmark et al. (2003). The code is executed each sampling period `h`.

```
y = readY();
ref = readReference();
e = ref - y;
v = K*e + I;
```

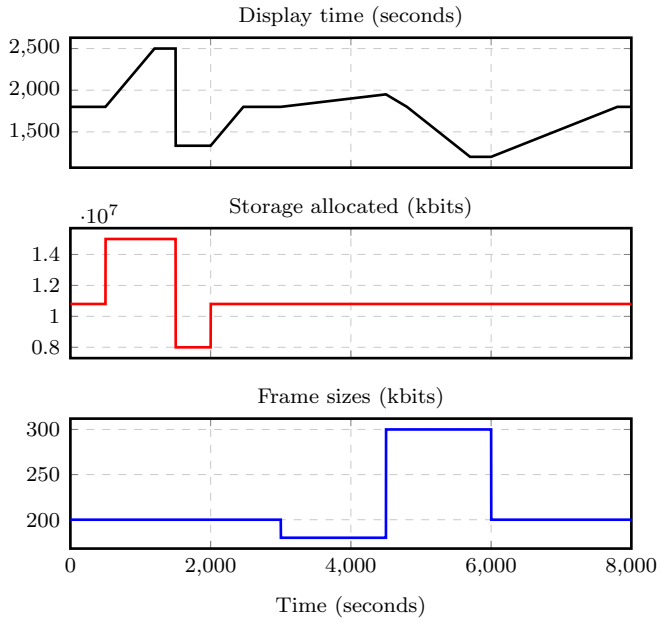

Fig. 3. Stored video time, allocated storage, and frame size in open loop.

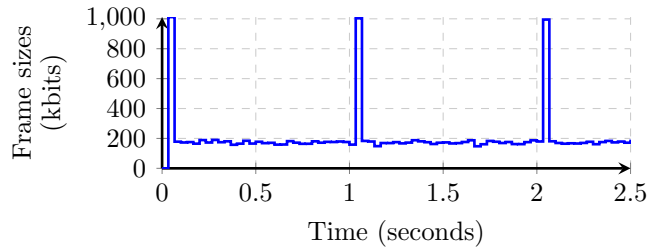

Fig. 5. Frame size detail in closed loop.

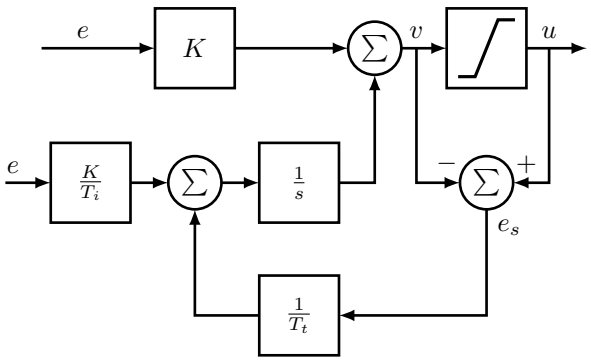

Fig. 6. A PI controller with tracking-based anti-windup.

```
u = max(u_low, min(v, u_max));
writeU(u);
I = I + (K/Ti)*e + (h/Tt)*(u-v);
```

If the tracking time constant T_t is selected as $T_t = h$ then the reset will be performed instantaneously. This is also known as *deadbeat tracking*.

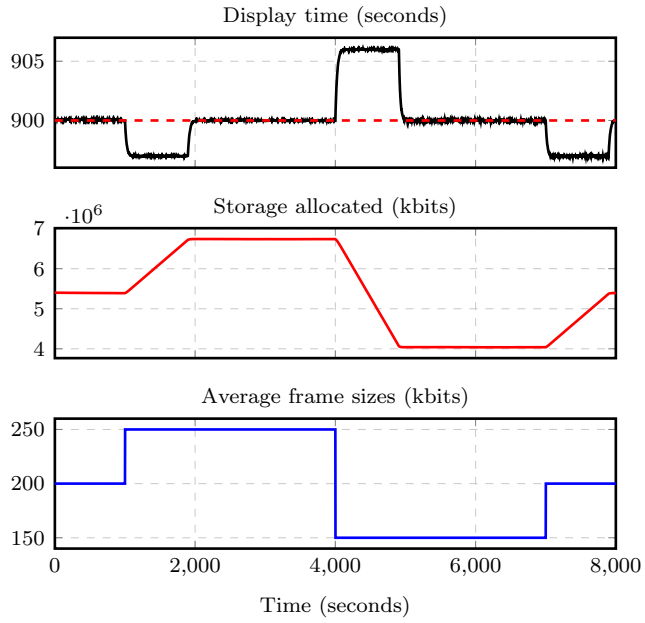

Fig. 4. Average frame size, target storage and buffer behavior in closed loop (K=5000, Ti=0.002).

4. TRACKING FOR HANDLING GLOBAL RESOURCE CONSTRAINTS

The proposed method is based on calculating the sum of the control signals that the individual controllers would like to send out and compare this value with $U_{\max}$, i.e., the maximum amount available. The difference between these values can be viewed as a tracking error signal, e_g, that is fed back to each individual controller through the gains K_T/ω_i where K_T is used to scale the gains and ω_i is a weight (or priority) that gives control over the relative importance of the controllers, i.e., which controllers that should be affected the least and the most by the lack of resources.

The rational behind the method is that as long as the sum of the control signals is larger than $U_{\max}$ then the error will be negative and this will cause the integral parts in all the controllers to decrease until eventually the sum of the control signals equals $U_{\max}$. The individual rates at which this takes place is controlled by ω_i. A small value of ω_i will make the gain large. Hence, the rate at which the integrator is adjusted will be large. A large value of ω_i will make the gain small and, hence, the rate at which the integrator is adjusted will be small. The result of this is that the control loops with large weights will be affected less by the lack of resources compared to those with small weights.

The proposed method is shown in Fig. 7 for the case of two controllers. The global tracking feedback loops are shown in red. The lower limit in the global saturation block can be set to zero and the upper limit is set to $U_{\max}$. The local saturation blocks could also have a lower limit of zero and the upper limit can be used to further constrain the value of the control signal.

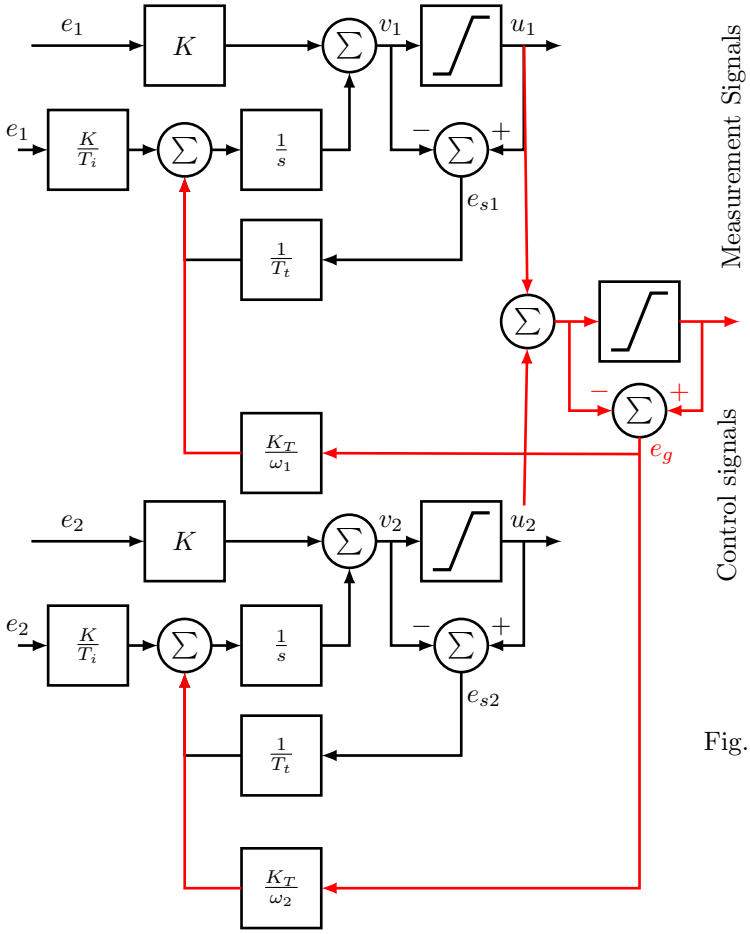

Fig. 7. A tracking-based approach for handling global control signal constraints. The figure shows the case for two local controllers.

4.1 A simple example

As a simple example of the approach we use three PI-controllers that each control a first-order linear system given by $G_p(s) = 1/(s+1)$. However, note that it is not a model of the storage system used in the simulations in 5, it is only intended as an example of the new approach for handling global control signal constraints.

The PI parameters are equal for the three controllers ($K = 2$ and $T_i = 1$) and the set-points are chosen as 10 for all three loops. Since the closed loop systems have static gains 1, the control signals will be equal to the reference values as shown in Fig. 8. Here also the sum of the control signals is shown (equal to 30) and $U_{\max}$ which in this case is set to 40. Hence, in the first part of the plots the global tracking is not invoked.

However, at time $t = 6$, we lower $U_{\max}$ to 20. Now the amount of available resources is less than what is needed. All the controllers have the same weight, i.e., they will share the available resources in a fair way and all will obtain the control signal equal to $20/3 = 6.67$. Finally, at time $t = 12$, we change the weights so that controller 1 has highest weight, and controller 3 lowest weight. This will change the control signals. Controller 1 will be affected

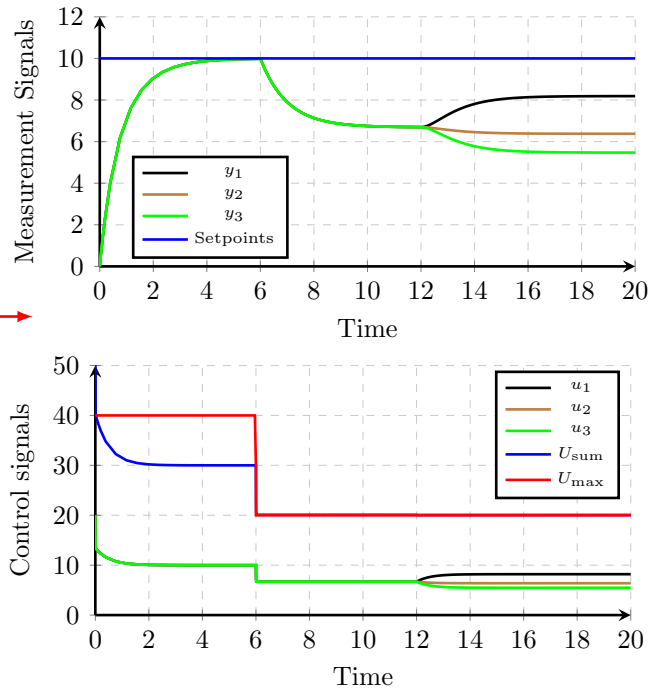

Fig. 8. Three identical controllers which each has a control signal equal to 10. The upper plot shows the set-points (all equal) and the measurement signals whereas the lower plot shows the control signals, the sum of the control signals (in blue) and $U_{\max}$ in red. At time 6 the resource constraint is activated and at time 12 the relative weights of the controllers are changed.

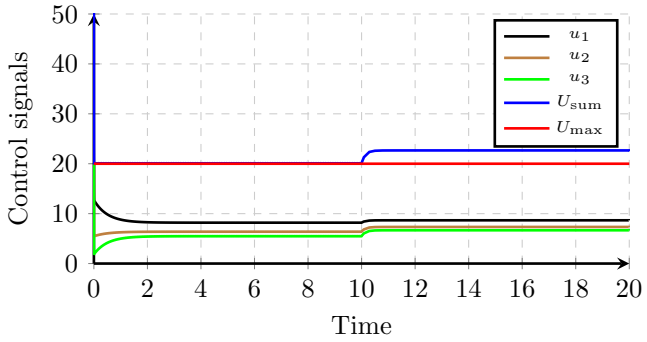

Fig. 9. The same example as in Fig. 8 with only the control signal information shown. At time 10, K_T is changed from 100 to 1.

the least by the resource constraint and controller 3 the most.

As shown by the example the proposed approach will adjust the control signals of all the involved controllers. In some cases this may be undesirable and one would prefer to only adjust a subset of the controllers, e.g., one might want to put the full adjustment on the control loop with lowest priority if that is possible and, if not, then continue with the loop with the second-lowest priority. This would require that the tracking is done instantaneously in a similar way as deadbeat tracking in ordinary anti-windup tracking. Another extension could be to use dynamic priorities instead of static. For example one could let the

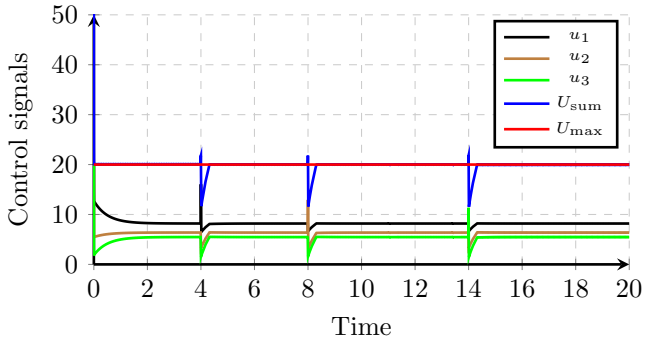

Fig. 10. The same example as in Fig. 8 now with $K_T = 100$ all the time but with input pulse disturbances.

adjustment on the control loops depend on how much more resources they need than what is available.

The proposed method, however, also has issues. In the previous figure, the parameters were $K_T = 100$, $\omega_1 = 1$, and $\omega_2 = 0.5$ and $\omega_3 = 0.4$, i.e., the corresponding tracking time constants are substantially smaller than the time constants of the closed loop systems. If we, however, change this by setting $K_T = 1$ then the system will converge to another, undesired, equilibrium where the tracking error e_g is different from zero and, consequently, the control signal constraint is violated, see Fig. 9 where the change occurs at time $t = 10$. This equilibrium occurs when $\forall k$, $(K_k/T_{Ik})e_k + (K_T/\omega_k)e_g = 0$, i.e., the total input to all the integral parts equal 0, while $e_g \neq 0$.

In addition to this, it is possible for the control signal constraint to be violated due to high-frequency disturbances. An example of this is shown in Fig. 10 where $K_T = 100$ all the time but pulse disturbances are introduced at the input to the plants at time $4, 8$, and 14. This disturbance again causes the control signal constraint to temporarily be violated.

5. RESULTS

We consider three cameras with similar (but independently generated) frame sizes but different duration set-points and a global constraint on the amount of resources available. The simulation is started in steady state (the cameras have filled their storage and reached their duration set-point). At time t = 1000s the average frame size of camera 1 increases from 200 kbits to 250 kbits, followed shortly after by camera 2 and then camera 3. At time t = 4000s the average frame size drops down to 150 kbits until t = 7000s where it goes back to 200 kbits. We number the different phases to ease understanding (phase 1 is from $t = 0$ to 1000, phase 2 from $t = 1000$ to 4000 ...)

In the first simulation (Fig. 11), the feedback tracking gains have been set to the the same value (100), i.e., each camera has same priority and should react equally to the global constraint. In this simulation both I and P/B frames are included according to Fig. 5 although in the plots only the average frame size is shown. At phase 2 the rise in frame size triggers an increase of storage until the global maximum is reached. From this point the global storage amount does not increase anymore due to the constraint and, thus, the duration of video stored for each camera drops until reaching the new equilibrium. When the frame sizes decreases again in phase 3, the amount of storage needed decreases and, thus, the duration increases until the duration set-point is reached. From this point, the storage required decreases. At phase 4, the average frame size goes back to 200 kbits and storage rises to compensate, keeping the defined set-point. We can see in this example that the proposed system works well.

The second simulation (Fig. 12) presents the same behavior as in the first one but here the cameras have different tracking gains. In this simulation we can see that the global behavior is very much like the one in (Fig. 11) but the durations are different due to that the global constraint is affecting different cameras more or less depending on the their tracking gain. Camera 3 with the lowest priority (highest tracking gain) will suffer the most from the limited resources and camera 1 with the highest priority (smallest tracking gain) will be affected the least. also this behavior corresponds to what could be expected.

In the simulations the basic version of the approach without any PI-tracking is used. The maximum value of the constraint violation is 0.04 per cent, which easily could be managed by adding a safety margin to the global constraint. It could also be reduced by increasing K_T.

6. EXTENSIONS

The problems mentioned in Section 4.1 motivate extensions to the proposed approach. We present here two (partly overlapping) extensions:

(1) Safety Margin, and
(2) PI-based Tracking.

The easiest way to address the problems above is to introduce a safety margin, i.e., to use an $U_{\max}$ that is smaller than the true resource constraint. The problem with this approach is that it still does not provide any guarantees that the resource constraint will be met. However, in practice this can work quite well.

The background for the the second extension is the risk of ending up in the undesired equilibrium discussed in the previous sub-section and shown in Fig. 9, i.e., where $e_g \neq 0$. In an ordinary control loop, the remedy to remove stationary errors is to introduce integral action. This can be used also at the global tracking level, i.e., by introducing a PI-controller that aims to remove the global tracking error e_g. The approach is illustrated in Fig. 13. The input to the PI controller is the original global tracking error and the set-point is 0, i.e., we want the PI controller to ensure that global tracking error really is zero. The output of the PI controller is connected to the K_T/ω_i blocks in the same ways as the global tracking error was in the case without the additional PI controller.

Using this approach it is possible to remove the stationary error as seen in Fig. 14. Here the same setup as in Fig. 9 is used, i.e., with $K_T = 1$. From $t = 0$ to $t = 10$ the ordinary global tracking approach is used. At $t = 10$ a properly tuned PI controller according to Fig. 13 is activated and the stationary error is removed. However, also in this case the global control signal constraint can be violated due to measurement noise.

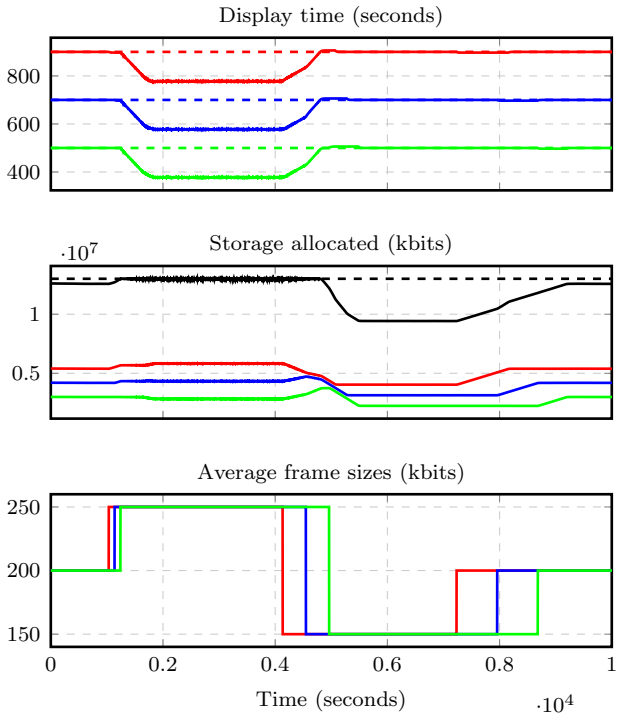

Fig. 11. Stored video time, allocated storage, and average frame size. The weights of the control loops are identical. The red signals are for Camera 1, the blue for Camera 2 and the green for Camera 3. The sum of the control signals is shown in black and the constraint in dashed black.

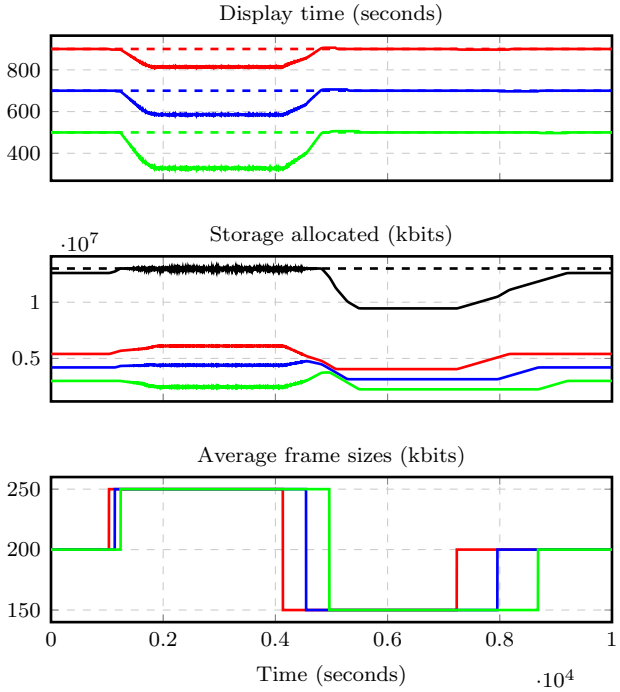

Fig. 12. Stored video time, allocated storage, and average frame size with different weights ($\omega_1 = 100$, $\omega_2 = 133.3$, $\omega_3 = 200$). The red curves are for Camera 1, the blue for Camera 2 and the green for Camera 3. The sum of the control signals is shown in black and the constraint in dashed black.

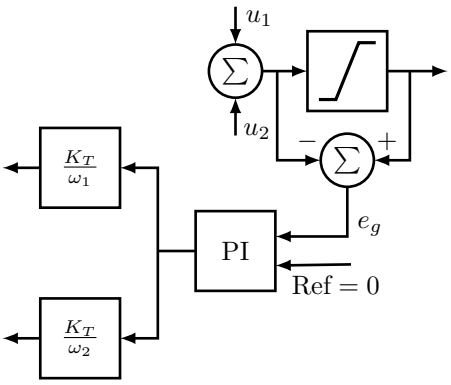

Fig. 13. PI-based Tracking. The blocks should replace the right hand part of the block diagram in Fig. 7.

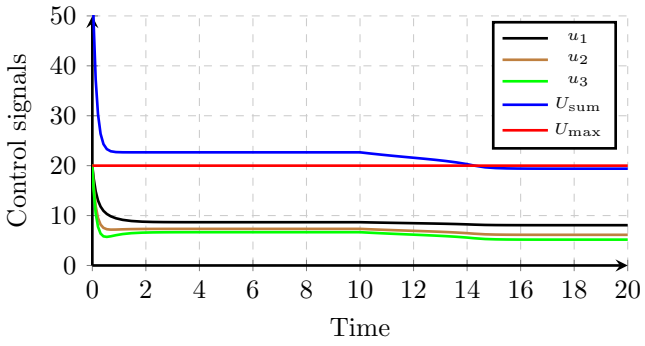

Fig. 14. The same example as in Fig. 9 with only the control signal information shown. At time 10, PI-based global tracking is activated.

7. CONCLUSIONS AND FUTURE WORK

A method for enforcing soft resource constraints for the case when the constraint is expressed as a global limitation on the sum of the control signals has been proposed. The method is inspired by tracking-based anti-windup for PID control. It has been applied to storage of video stream generated by surveillance cameras. This problem can be modelled as a set of control loops where each controller decides how much disk space is available for the corresponding camera. The approach has been evaluated in simulation with very good results.

The proposed method and its application to video storage can be continued and further extended in a number of ways. Concerning the method the following are possible future directions. The formal properties of the approach need further study to, e.g., analyze the multiple equilibria that may occur. The approach can also be modified in different ways. The approach could be applied to controllers without any integral part. In that case the global tracking signal could instead be added to the control signal v. One could also consider to instead add the global tracking signal to the set-point, e.g.,, to reduce the set-point in case of resource shortage. It is also likely that the approach can be used for certain other types of global constraints,

e.g., on the process outputs. Finally, the use of dynamic priorities needs to be further explored.

For the video storage application the most natural next step is to implement it on physical storage and use real video streams. Another possibility is to also include the shared communication resource and use the potential that the cameras have for adapting the compression rate and the frame rate. One limitation that has not been considered in this paper is the delay between the storage set-point request and allocation which could result in a control limitation. This should be introduced and studied. The communication bandwidth is also a shared constrained resource that interacts with the storage resource in interesting ways. For example, if a camera does not receive enough storage resources then it does not need so much bandwidth and, consequently, if it does not receive sufficient bandwidth then it does not require as much storage. A combined control approach for this is a challenging goal.

REFERENCES

Astrom, K.J. and Murray, R.M. (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press.

Cucinotta, T., Palopoli, L., and Marzario, L. (2004). Stochastic feedback-based control of qos in soft real-time systems. In *IEEE Conference on Decision and Control*, 3533–3538 Vol.4.

Cucinotta, T., Abeni, L., Palopoli, L., and Lipari, G. (2011). A robust mechanism for adaptive scheduling of multimedia applications. *ACM Trans. Embedded Comput. Syst.*, 10(4), 46:1–46:24.

Cucinotta, T., Lipari, G., Palopoli, L., Abeni, L., and Santos, R.M. (2009). Multi-level feedback control for quality of service management. In *IEEE International Conference on Emerging Technologies and Factory Automation*, 1–8.

De Cicco, L., Mascolo, S., and Palmisano, V. (2011). Feedback control for adaptive live video streaming. In *ACM Conference on Multimedia Systems*.

Edpalm, V., Martins, A., Årzén, K.E., and Maggio, M. (2018). Camera networks dimensioning and scheduling with quasi worst-case transmission time. In *Euromicro Conference on Real-Time Systems*, 17:1–17:22.

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

Leva, A. and Maggio, M. (2010). Feedback process scheduling with simple discrete-time control structures. *IET Control Theory Applications*, 4(11), 2331–2342.

Maggio, M., Hoffmann, H., Santambrogio, M.D., Agarwal, A., and Leva, A. (2010). Controlling software applications via resource allocation within the heartbeats framework. In *IEEE Conference on Decision and Control*, 3736–3741.

Palopoli, L., Abeni, L., Cucinotta, T., Lipari, G., and Baruah, S.K. (2008). Weighted feedback reclaiming for multimedia applications. In *IEEE/ACM Workshop on Embedded Systems for Real-Time Multimedia*, 121–126.

Palopoli, L., Cucinotta, T., Marzario, L., and Lipari, G. (2009). Aquosa - adaptive quality of service architecture. *Softw., Pract. Exper.*, 39(1), 1–31.

Rawlings, J. and Mayne, D. (2009). *Model Predictive Control: Theory and Design*. Nob Hill Pub.

Richardson, I.E. (2010). *The H.264 Advanced Video Compression Standard*. Wiley Publishing, 2nd edition.

Wittenmark, B., Åstrom, K., and Årzén, K.E. (2003). Computer control: An overview. Technical report, Department of Automatic Control, Lund University.

Yin, X., Jindal, A., Sekar, V., and Sinopoli, B. (2015). A control-theoretic approach for dynamic adaptive video streaming over http. In *ACM Conference on Special Interest Group on Data Communication*, 325–338.