Dynamic Management of Multiple Resources in Camera Surveillance Systems

Martins, Alexandre; Årzén, Karl-Erik

Published in:
IEEE Access

2021

Dynamic Management of Multiple Resources in Camera Surveillance Systems*

Alexandre Martins[1] and Karl-Erik Årzén[2]

***Abstract*— Distributed camera surveillance systems typically consist of multiple cameras that need to store some fraction of their video streams in a central storage node. The disk space of this node as well as the network between the cameras and this central node constitute shared resources. In the paper the disk space allocation as well as the network bandwidth reservation are solved using techniques normally associated with process control. These include mid-range control and tracking-based control of global shared resources. The approach is evaluated by simulations.**

I. Introduction

Computer and communication systems are interesting and challenging application areas for control, e.g., [16]. However, in many applications, control is used only for individual control loops. A large-scale system, e.g., a camera surveillance system, contains multiple subsystems at different levels that interact in often very challenging ways. This can be compared with large-scale industrial production processes where control is successfully used as a core technology for guaranteeing performance and stability, and where a number of techniques have been developed for combining multiple individual controllers in order to fulfil different global objectives. One of the aims of this paper is to show that these ideas also apply to computer and communication systems, with camera surveillance systems as the particular case.

A common scenario is the need to share some resource among a number of clients or subsystems. In these cases a limited shared resource, e.g., communication bandwidth, CPU capacity, or memory, should be shared between a number of clients (tasks, processes, nodes, ...). In many cases, the amount of resource that should be allocated is given by the output of a controller, i.e., by a control signal, which has the objective to keep some quality or a performance related variable at the setpoint. Hence, the overall architecture consists of a number of control loops that interact with each other through the fact that the sum of the control signals, i.e., the total amount of resources required, is limited. In our previous paper [13] an approach for managing shared resources inspired by the tracking or back-calculation method for handling anti-windup in PID control, was proposed as a way of handling shared disk space resources in video storage systems. The approach was developed to fit well with a PID-based solution, which is often sufficient for these types of systems that are often of low order and very decentralized, as opposed to, e.g., a conventional MPC approach that also could be used to handle this type of control signal constraints. The price to pay for decentralization is that the global control signal constraint must be soft in nature. Here this approach is extended to simultaneously manage two dependent resources: shared disk space and shared communication bandwidth. The underlying control architecture is based on cascaded PID control and mid-ranging.

Video surveillance systems that are increasingly prevalent in society are used at different levels and scales – e.g., cities, public places, companies, homes, etc. A typical system comprises multiple (in some cases several hundreds or thousands) cameras, disseminated over an area and recording 24/7. Today, the video industry focuses on IP cameras that stream videos that are compressed using the H.264 standard [17], 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 shared network to one or several storage stations, where the video streams are monitored by a human operator either in real-time (”live”) or by inspecting stored video sequences. The former requires that the latency of the video stream is not too long. The latter requires the operator to have access to a sliding window of the stored 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 and rate of the 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 the cameras compete for, see Fig. 1. Similarly, transmitting the video streams from the cameras to the storage station requires network bandwidth that also constitutes a shared resource.

Both allocation of storage and allocation of network bandwidth can be viewed as control problems. In the storage case, the measured variable is the duration of the stored video window. In the sequel this is also referred to as the ***retention***. This is compared to the desired duration and the error is fed to a controller that calculates the amount of memory or disk space that the video stream may use and/or the video compression level. The frames in the video stream and the storage that they require can be viewed as a disturbance acting on the system. In the network case, the measured variable is the latency that is compared to the desired latency. The delay can be affected either by changing the share of the total network bandwidth that is allocated to the stream or by changing the video compression level. High compression

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).

[1] is with Axis Communications and with Department of Automatic Control, Lund University, Sweden

[2] is with Department of Automatic Control, Lund University, Sweden

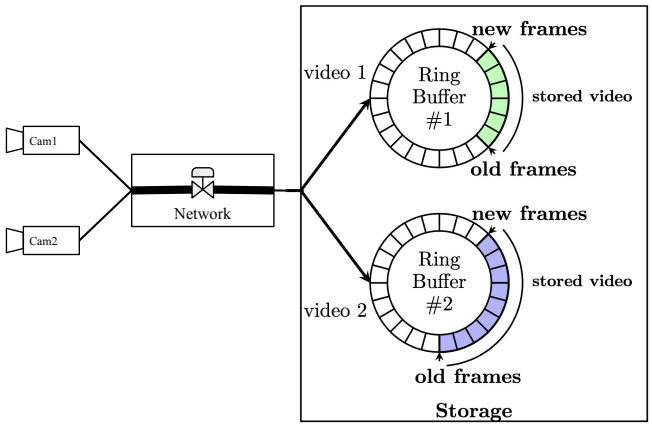

Fig. 1. System overview.

means that the video frames are smaller and hence, faster to transmit. Changing the video compression also influences the storage allocation – a video stream with high compression requires less disk space than a video with low compression. Hence, the two control problems are coupled.

The following are the contributions of this paper.

- The allocation of disk space and network bandwidth for a video stream are formulated as control problems.
- A decentralized way to manage and prioritize constrained resources is proposed.
- It is shown how cascaded PID controllers and mid-ranging in combination with the tracking-based approach for managing shared resources from [13] can be used to control two dependent shared resources.

A. Related Work

Control has been applied to the problem of determining the best setting for video streaming [6, 5, 14, 18]. 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 [3] or multimedia [15, 4] application. The issue 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 to avoid exceeding the amount of available storage.

The general formulation of our problem is allocating a limited resource to multiple actors. This problem has been encountered in different circumstances with controlling computing systems, e.g., in the management of a set of threads [10], CPU scheduling [11], or core allocation [12]. However, the solutions that were found either require domain knowledge or do not scale well. We aim to provide 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 and network controllers. The objective has been an approach that is as decentralized as possible and fits well into a PID control setting.

B. Outline of the paper

Camera surveillance systems and how they have been modeled in this paper are discussed in Section II. The control system architecture including the network bandwidth and storage controllers are presented in Section III. Simulated evaluation results are given in Section IV. Finally, conclusions and suggestions for future work are discussed in Section V.

II. Camera Surveillance Systems

A typical surveillance system consists of multiple network video cameras that stream continuous video 24/7 to a central storage which can be local or remote. The video streams are encoded using the H.264 video compression standard (see [17]). H.264 is a block-oriented, motion-compensated integer-DCT coding technique. A H.264 video consists of a sequence of groups of pictures (GOP). Each GOP comprises 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. The size of typical H.264 surveillance video frames (and by extension the bandwidth) can be predicted using the model reported in [7] and developed in detail in [8]. An example of the expected average video frame size for a defined sets of environment parameters is shown in Fig. 2.

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. 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. In this paper we only consider the first case: recycling for memory re-use. The storage cost has 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.

A common limitation to the amount of video stored is the amount of bandwidth available on the network infrastructure to deliver the video to the storage endpoint. The network infrastructure can be be tailored for the video surveillance system or, as we see more and more frequently, tailored for a larger use case and shared among other applications. This brings a second dimension to the video storage problem as we need to be able to stream the video to the end point in a defined amount of time to avoid saturation. Too high bandwidth constraints on the network infrastructure will delay or possibly cause data packets drops, leading to high video delay and/or video loss. This problem is even more important when the video camera system is used for live viewing as well as recording. This requires a short transmission delay to be able to react based on the video content.

In our work, we consider the two dimensions of the problem: we are given a fixed amount of available global

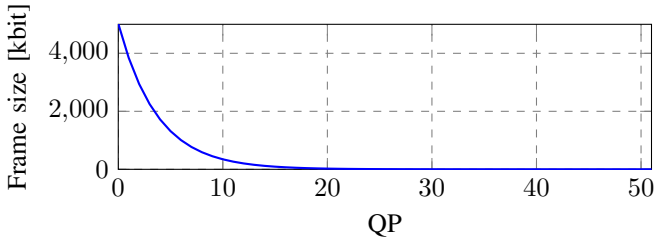

Fig. 2. H.264 average video frame size for different QP (compression) values.

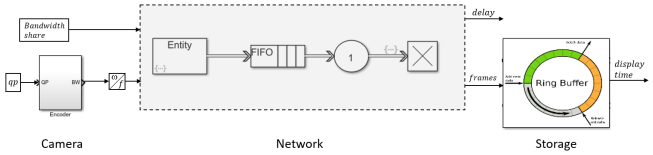

Fig. 3. The overall structure of the model.

storage, i.e., disk space and we want to optimize its usage. We also would like to limit the delay of the transmission of the video over the network. We want to keep as much video as possible, satisfying given quality constraints, while ensuring a bounded transmission delay. This can be made easier by selecting which constraint is the most important for each camera. For a camera used for live viewing, a low delivery delay is crucial, while a video dedicated to storage applications could accept a higher delay but has higher storage requirements.

A. Model of the camera surveillance system

The model consists of three parts: the camera model, the network model, and the storage model, see Fig. 3. The model is implemented in Simulink and partly makes use of Simulink's discrete event simulation toolbox SimEvent (as used for example in [9]).

The camera model consists of the encoder based on the model in [7]. It calculates the bandwidth, ω, of the video stream as a function of the compression level, qp. A number of scene and camera parameters can be given to model how the bandwidth depends on, e.g., variations in the scene caused by changing light, movements and camera/encoder capabilities. Assuming a fixed frame rate the bandwidth can be converted into frame size, f. The cameras that we consider have 52 discrete compression levels where 0 means lossless compression and 51 maximum compression. In the model, we, however, assume that the compression levels are continuous. The relationship between the compression and the frame size is highly nonlinear, see Fig. 2.

The network model assumes an ideal reservation-based network where each camera is assigned a separate network channel with a channel bandwidth that constitutes a given share, or percentage, of the total network bandwidth. These channels are used to transmit the video stream from the cameras to the storage without any interference between the channels. It is also assumed that a small percentage of the total network is allocated to a special control plane channel used for exchanging control and synchronization messages. These messages are, however, not explicitly modeled and

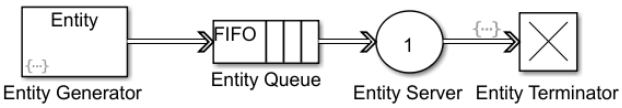

Fig. 4. The SimEvent model of a network channel.

are assumed to take zero time to transmit (which is actually quite realistic since they are very small compared to the video frames).

The channels are modeled in a rather simplistic way. It is assumed that the network channel share is an input to the channel model and at any given moment the sum of the shares of all the camera channels plus the control plane channel should be less than 1, i.e., the total network bandwidth is a shared resource between the cameras.

Each channel is modelled by the discrete sequence of frames that are queued in FIFO (first-in first-out) order and eventually transmitted. The inputs to the channel model are the frame size and the channel share. The outputs are the frame delay measured as the difference in time between when the frame enters the FIFO and when it has been transmitted and arrives at the storage.

The frame model is implemented using SimEvent, see Fig. 4. Frames and the corresponding SimEvent entities are created at a rate given by the camera frame rate and with the current time and the frame size as attributes. The FIFO (first-in first-out) entity queue has an infinite buffer, i.e., no frames are lost due to buffer overflow. The entity server consists of one server for which the service time, i.e., the transmission time of the frame, is calculated as the frame size divided by the channel bandwidth. When the frame arrives at the entity terminator the delay since the frame entered the FIFO is calculated.

An ideal reservation-based network can be approximated by a TDMA (time-division multiple access) network protocol where the sending slots are infinitely small and the schedule is sufficiently long to avoid quantization of the reservation shares (or budgets).

The storage model is the same as in [13]. A Simulink S-function is used to implement a ring buffer containing a sliding window of stored frames for each camera. The input of the model is the amount of disk space available, expressed as a share, or percentage, of the total amount of disk space. The output is the duration of the sliding window. If the amount of disk space increases, the stored video time will increase linearly as new frames are entered into the buffer until the buffer becomes full. If the amount of disk space is decreased, the stored video will drop instantaneously as the corresponding number of frames will be flushed from the buffer. If the average frame size is increased or decreased, the stored video time will decrease or increase as there will be room for less or more frames in the buffer. 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.

III. Control System Architecture

The control system architecture is based on a combination of cascaded PI controllers, mid-ranging, and the tracking-

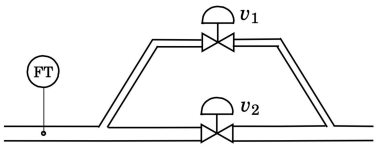

Fig. 5. Mid-range application example.

based approach to handling shared resources proposed in [13]. The PI controllers are all discretized versions of the standard PI controller given by

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

A. Feedback linearization and cascade control

In the camera model the encoder is highly nonlinear. Since we can measure the bandwidth generated by the encoder we use a PI-controller – the ***camera bandwidth controller (CBC)*** - to help linearize it. The measured variable is the generated camera bandwidth which is then compared to the bandwidth setpoint. The output of this controller is the compression level of the encoder. Hence, instead of specifying the compression level of the camera one specifies the desired output bandwidth.

The camera bandwidth controller is the inner loop in a cascade control structure where the outer loop controls the delay of the video stream by adjusting the camera bandwidth. The outer controller is also a PI-controller which we denote the ***delay bandwidth controller (DBC)***. Alternatively, if the main control objective is to control the stored display time, then the outer loop is a PI-controller that controls the stored display time by adjusting the camera bandwidth. We call this controller the ***retention bandwidth controller (RBC).***

B. Mid-range control

Cascade control is a strategy where one control signal and two measurement signals are used to meet the control objective. The dual situation is when two control signals are used to control one measurement signal. This situation occurs twice here. The control system has two actuators for controlling the delay: the cameras' share of the total network bandwidth and the camera's bandwidth (or compression level). Similarly there are two actuators for controlling the stored display time: the disk space allocated to the ring buffer and the camera bandwidth. The question then is how to combine these two actuators in the best way.

In process control, mid-range control or simply mid-ranging, is commonly used for this [1]. The motivating example contains two valves that are used to control one flow according to Fig. 5. Valve v_1 is small, i.e., has low control authority, but has high resolution whereas valve v_2 is large, i.e., has high control authority, but low resolution. The solution is to use valve v_1 to control the flow and then use a so called *valve position controller (VPC)* that uses the control signal of v_1 as the measurement signal, and v_2 to ensure that the control signal of v_1 lies in the middle of its operating range, e.g., 50%. The control structure is shown in Fig. 6. Process P_2 and controller C_2 together form a fast feedback loop. The mid-ranging controller C_1 controls the valve position of controller C_2 via the process output y.

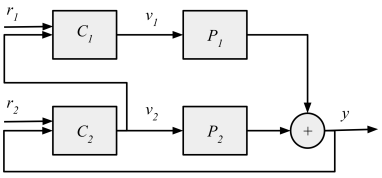

Fig. 6. Mid-range controller structure.

In the delay control case, the channel bandwidth, i.e., the channel's share of the total network bandwidth, corresponds to the small valve. It is a fast actuator that directly influences the delay but the operating range is limited since the network bandwidth is shared among all cameras. The frame size (or alternatively the camera bandwidth) actuator is slow since a change in frame size will not have any influence on the delay until all the currently queued frames have been transmitted. The limited number of compression levels also reduce the resolution. Hence, the frame size actuator corresponds to the large valve. The control system then consists of two controllers. One uses the delay as the measurement signal and the channel bandwidth as the control signal. We call this controller the ***delay channel controller (DCC)***. The second uses the control signal of the first as the measurement signal and the camera bandwidth as the control signal. This is the previously mentioned ***DBC***. Thus both are PI-controllers. The setpoint of the latter controller should be 50% of the operating range of the controller, i.e., 0.25, in case of two identical cameras and $1/6$ in case of three identical cameras. This choice will maximize the control authority of the ***DCC***.

Mid-range control is also used for the storage. The small actuator is the share of the total storage amount and the large actuator is, again, the frame size (or alternatively the camera bandwidth). The ***retention storage controller (RSC)*** uses the stored video duration as the measurement signal and the share of the total storage amount as the control signal, and the ***RBC*** uses the control signal of the ***RSC*** as the measurement signal and the camera bandwidth as the control signal.

An example of how the mid-range control works for the delay control is shown in Fig. 7. The top plot shows the delay with and without mid-ranging. The bottom plot shows the corresponding control signals. At $t = 30$ the setpoint changes from 0.2 to 0.1. Without mid-ranging the delay channel controller compensates for this by increasing the channel's share of the total network bandwidth from 0.5 to 1.0. With mid-ranging the delay bandwidth controller ensures that the control signal returns to 0.5 by increasing the compression level. The small undershoot that is visible can be removed if also a feed-forward compensator is introduced between the two controllers, see [1] pg 379. This has, however, not been considered necessary in our case.

C. Tracking-based resource management

The approach that we use to handle the two shared resources is the tracking-based resource sharing approach first introduced in [13]. The extension here is the use of this approach for two dependent resources and their interaction with the mid-ranging. The method is inspired by the tracking approach for handling anti-windup in controllers with integral action that is commonly used in PID control ([2]). It is assumed that the amount of shared resource that should

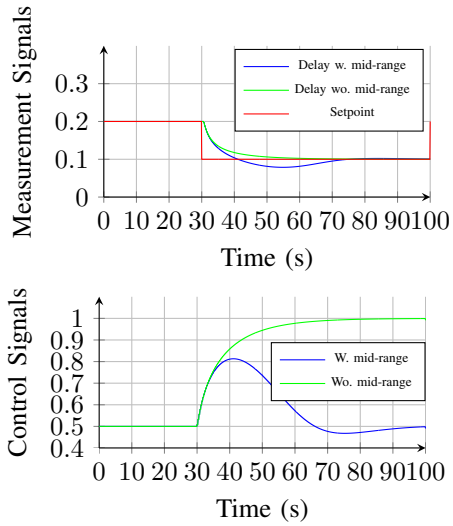

Fig. 7. Response to a setpoint change in the delay without (wo.) and with (w.) mid-range control.

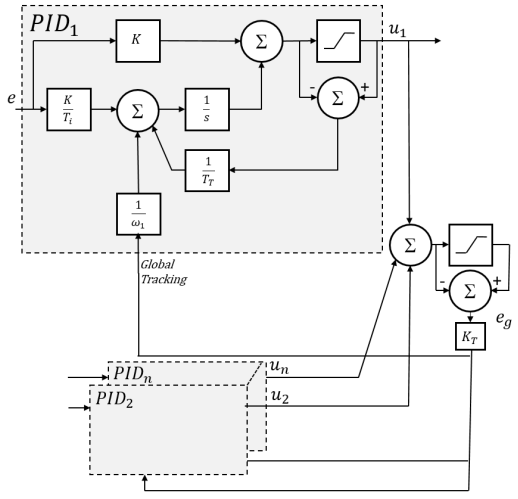

Fig. 8. Tracking-based resource sharing.

be allocated is given by the output of a controller. Hence, the overall architecture consists of a number of control loops that interact with each other since the sum of all the control signals, i.e., the total amount of resources required, is limited.

The proposed method, see Fig. 8, is based on calculating the sum of the control signals that the individual controllers would like to send out and by comparing this value with $U_{\max}$, i.e., the maximum amount available. The difference between these values can be viewed as an 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 uniformly, and ω_i is a weight (or priority) that gives control over the relative importance of the controllers, i.e., which controllers should be affected the least and the most by the lack of resources.

As long as the sum of the control signals is larger than $U_{\max}$, the error will be negative and this will, if the tracking time constants are substantially smaller than the time constants of the closed loop systems, 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 are 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. It results that the controllers with large weights will be affected less by the lack of resources compared to those with small weights.

The global resource tracking method is used for both the network bandwidth and the storage disk space. In the first case the approach is applied to the ***DCCs*** and in the second case to the ***RSCs.***

D. Mode Handling

A problem with the controller structure described so far is that the ***DBC*** and the ***RBC*** both use the same actuator, i.e., the setpoint of the ***CBC***. To handle this, two camera-specific modes are introduced. In the live mode, it is assumed that an operator monitors a live video stream. In that case, delay control is favored and the ***RBC*** is simply disabled. This means that in this case, the full responsibility for controlling the stored display time is given to the ***RSC*** without any help of the ***RBC***.

Similarly, in the stored mode it is assumed that the operator inspects the stored video streams. In that case, the control of the stored display time is favored and the ***DBC*** is simply disabled. This means that in this case, the full responsibility for controlling the delay is given to the ***DCC***.

E. Dependent resources

The network bandwidth and the storage disk space are dependent resources, in the sense that if a camera does not receive enough disk space, it does not need as much network bandwidth and vice versa. One simple approach for handling this is to use the same relative priority ordering among the cameras for both resources, i.e., if a camera has high priority with respect to the network bandwidth, it should also have high priority for the storage disk space. Using mid-ranging, however, reduces the need for this. If a camera requires more disk space than what it can get, then the ***RBC*** will increase the compression level, i.e., reduce the channel network bandwidth, until the camera eventually meets its disk space requirement. Finally, the cameras in live mode should have lower storage priority than the cameras in stored mode.

F. Global Controller Structure

The global controller structure is shown in Fig.9 except for the global resource tracking. The controller contains anti-windup mechanisms not discussed here due to space constraints. All the controllers are discrete-time controllers with a sampling rate equal to the frame rate.

The proposed controller structure is very decentralized. The only centralized operations are the summations of the control signals, the limitation, and the multiplication with K_T which scales linearly with the number of cameras.

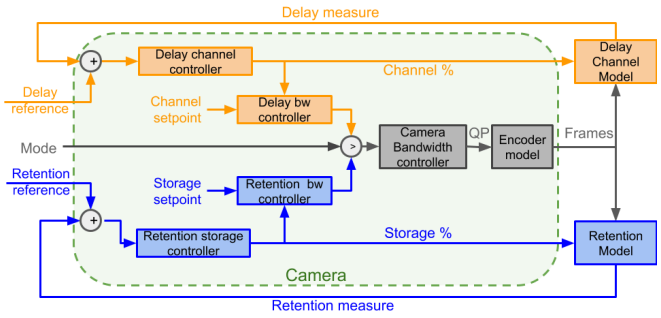

Fig. 9. A block diagram of the overall controller structure.

IV. Evaluation and Results

In order to evaluate the proposed approach four simulated scenarios are considered, the first three with two cameras and the last with three cameras. In the first three scenarios camera 1 (in blue) uses the live mode and camera 2 (in red) uses the stored mode. Both cameras have a fixed retention setpoint of 200 s and a delay setpoint of 0.2 s. The cameras have the same priority in the global tracking, both for the network channel bandwidth and the storage, in all scenarios but scenario 3 (only the camera mode will cause different behavior in case of resource saturation). In scenarios 2 and 3, a sampling period of 10 s for the storage and 5 s for the network bandwidth was used, this was not the case for the scenarios 1 and 4, where both used the cameras sampling period of 1/30 s.

Scenario 1: In the first scenario, a step disturbance in the average frame size, i.e. an offset, is added and there are no resource constraints. The result is shown in Fig 10. The second plot shows the resulting camera bandwidths and the third and fourth plot show the delay and the retention time. The retention plots at the beginning are not in steady state because the simulation is started with default initial parameters. We can see in the figure that the bandwidth changes for cameras 1 and 2 are slightly different. This is due to the bandwidth controllers, which are chosen based on the mode of each camera (the live camera uses the ***DBC*** while the other uses the ***RBC***). The change in the delay caused by the disturbance is smaller for camera 1 than for camera 2, as could be expected since camera 1 uses live mode. Similarly, the retention changes are much smaller for camera 2 than for camera 1.

Scenario 2: In the second scenario, resource constraints are introduced. At time $t = 500$ s, a network bandwidth saturation is introduced (and kept), and at time $t = 1000$ s, a global storage saturation is added (and kept). The priority for the cameras is the same, meaning that both cameras would react in the same way to the global resource tracking, based only on the mode it has (live or store). One can see in Fig 11, top two plots, that the limitation of total network bandwidth at $t = 500$ s triggers a delay increase and that both cameras correct their channel bandwidth accordingly. Camera 2 (store) is more impacted than camera 1 because it uses the ***RBC***, while camera 1 corrects the network channel as well as the bandwidth *via* the ***DBC*** to be able to regulate the delay more efficiently. In the bottom two plots one can see that the retention time for camera 1 (live) oscillates due

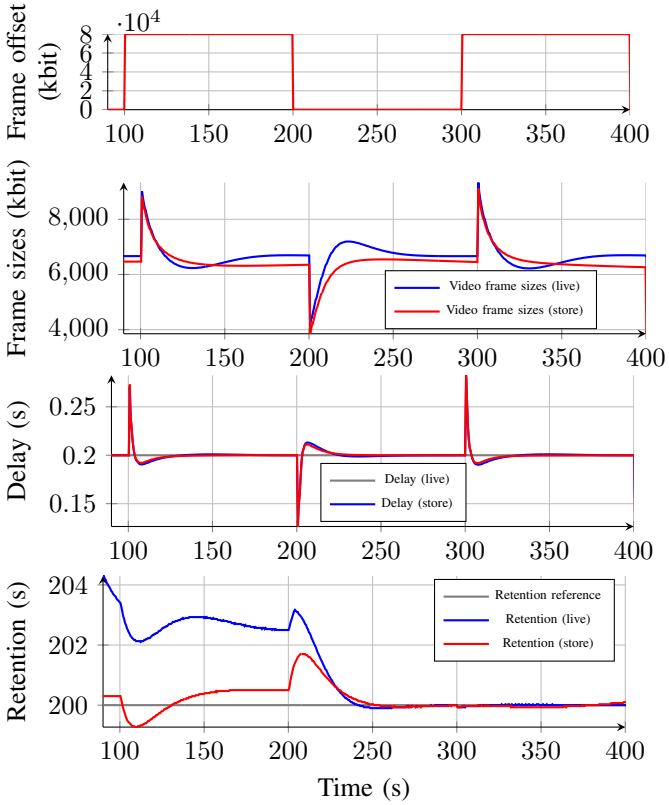

Fig. 10. Scenario 1: Delay and retention for two cameras with the same parameters. One uses live mode and the other store mode.

to the delay global saturation, because of the bandwidth adjustments to correct for the delay error. At time $t = 1000$ s, the global storage amount is decreased and triggers a drop of video data stored, which affects both cameras. Camera 2 being focused on video retention, it is less affected than camera 1. We observe that at time $t = 1000$ s, the delay of the camera 2 video rises. This is due to the camera being in retention mode and the bandwidth being controlled by the ***RBC***, which induces a higher bandwidth to compensate for the loss in retention, thus creating a higher delay in the video transmission.

Scenario 3: The third scenario is similar to the second one. The difference is that the cameras have different priorities. Camera 1 has a higher priority than camera 2 for both delay and retention. The plots and colors are the same as for scenario 2. One can see that during the saturation of the global network bandwidth and global storage, (Fig. 12 top two plots) camera 1 is less affected than camera 2 by the saturation of global resources availability. This scenario shows how the priorities can be used to optimize even more the system by giving priority to different cameras regarding the most critical resource needed.

Scenario 4: In the final scenario three cameras are used. The set-points are changed during the scenario and the frame size disturbance changes every 10 s. Resource limitations are also present. The modes of the cameras change over time but all the cameras have the same priority. One of them uses live mode and the others store mode. The camera using live mode

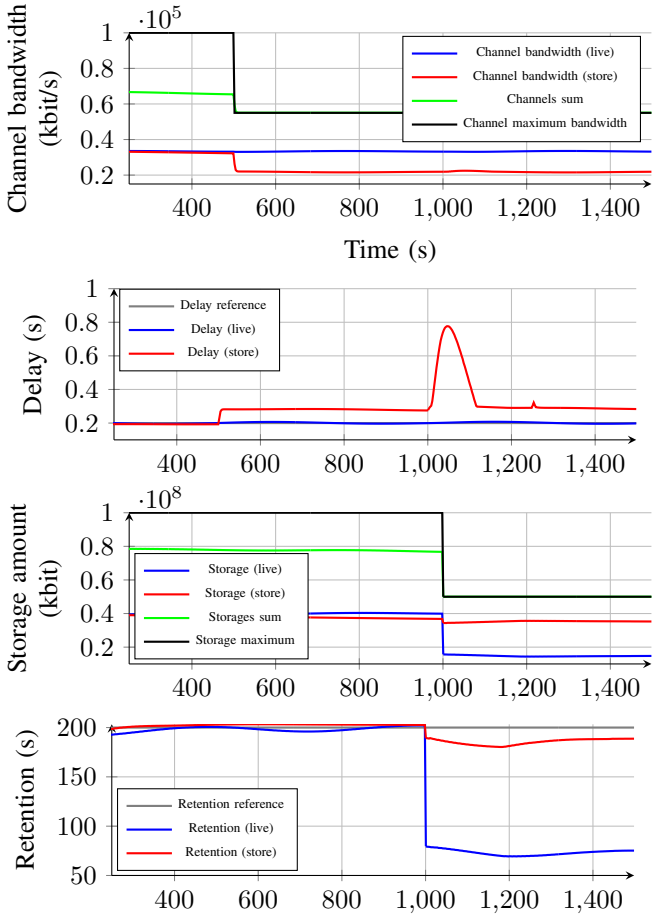

Fig. 11. Scenario 2: Delay and retention for two cameras with the same parameters with live and store mode, with and without global limitations.

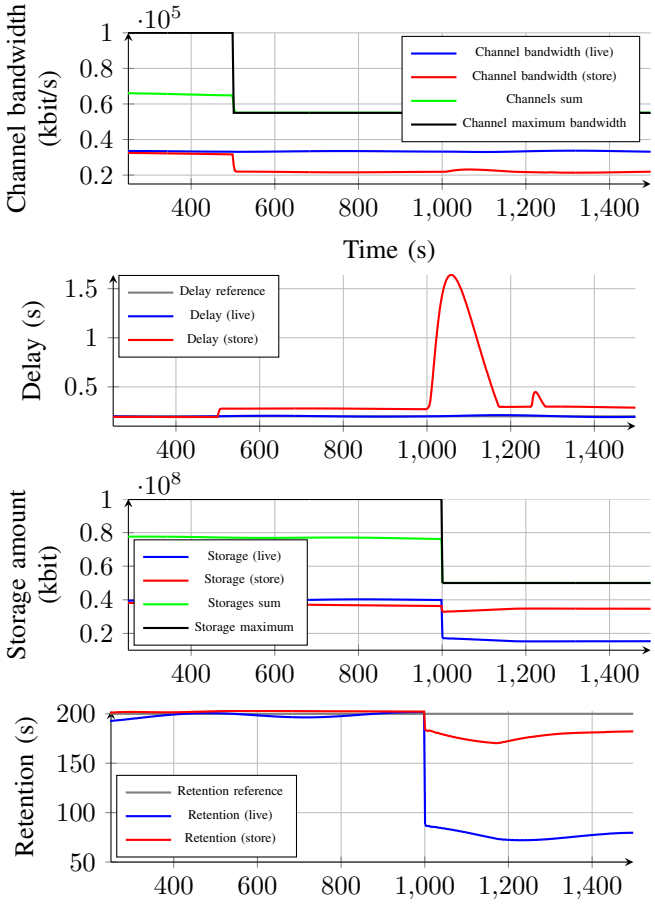

Fig. 12. Scenario 3: Delay and retention for two cameras with the same parameters with live and store mode, with and without global limitations and different global resource priority.

rotates over time (1,2,3,1,.... etc) every 500 seconds. This scenario is presented to show how the system would react when multiple disturbances and changes are occurring. It is closer to a real scenario where the frame sizes could change in a short time, and only one camera is viewed at a time.

One can notice the high frequency fluctuations due to the frame size disturbance in Fig. 13. These disturbances come from the difference in frame size in the simulated H.264 video. A H.264 video contains two types of frame, I-frame and P-frames, and there are usually around 30 P-frames for one I-frame. In our simulations, an I-frame is about 10 times the size of a P-frame.

One can also see that the set-points for delay and retention are followed in the periods where the global resource limitations do not limit the amount of resource available. When the global resources are limited, the system is still stable but cannot follow the set-points. One can further notice that the behavior of one of the cameras, both in the delay and retention part, is always different from the other two because it focuses on live streaming instead of storage. This demonstrates that, as expected, the mode of the camera influences which resource is prioritized. Note that in this scenario priorities of all the cameras are the same, meaning that they would react similarly to a global saturation of the resource, only the mode is changing their behavior regarding these limitations.

V. Conclusions and Future Work

In this paper we have proposed a control system architecture based on mid-ranging and global resource tracking that is used to control, assign, and prioritize resources in camera networks. This approach allows to address the allocation of discrete limited resources (network bandwidth and video storage) in a distributed multi tenant setting in order to meet multiple requirements (delay and retention time) by using common (video bandwidth) and separate actuators (network channel and global storage space).

A. Future Work

There are a number of future directions for this work. Using the available model, e.g, it is likely that performance can be further improved by introducing decoupling in the mid-range controllers. Additional changes should be introduced within controllers to handle saturation, and the saturation feedback could be directed to the bandwidth controllers to improve the saturation case. Also the network model should be made more realistic by, e.g., adding extra delay based on the total bandwidth. There are also additional limitations that need to be addressed before applying it to a real system. The difference in bandwidth and camera characteristics are bounded in the simulation. In a real scenario the cameras' characteristics (resolution, frame rate, lens, etc) and scene differences (motion, light, etc) would generate very different

bandwidth. Also the frame size changes are bounded. In reality the frame size changes a lot within a GOP, (I-frames are very large compared to P-frames) as well as due to motion or other scene changes, which could generate a high variability of sizes between consecutive frames. However, it is our belief that the proposed general controller structure would still be applicable.

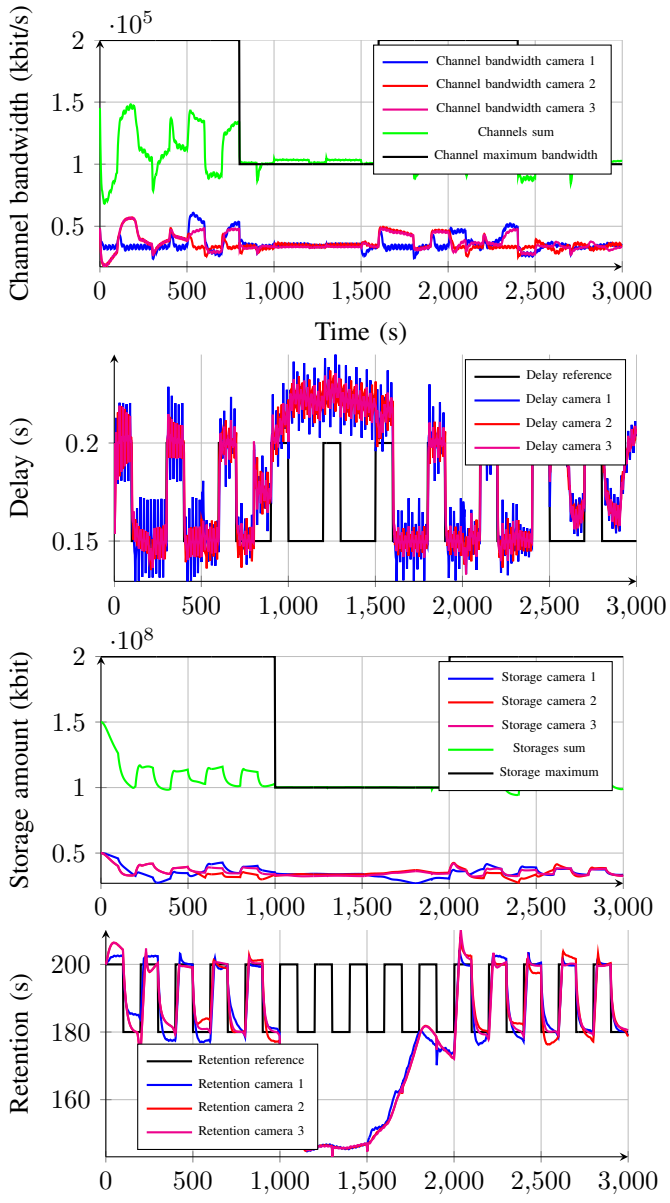

Fig. 13. Scenario 4: Dynamic scenario - frame disturbance, delay, network bandwidth, retention, storage and modes changes, priorities are the same

References

[1] Karl J Astrom and Tore Hagglund. *Advanced PID control*. Advanced PID control: ISA, 2006.

[2] Karl Johan Astrom and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[3] T. Cucinotta, L. Palopoli, and L. Marzario. "Stochastic feedback-based control of QoS in soft real-time systems". In: *IEEE Conference on Decision and Control*. 2004, 3533–3538 Vol.4.

[4] Tommaso Cucinotta et al. "A Robust Mechanism for Adaptive Scheduling of Multimedia Applications". In: *ACM Trans. Embedded Comput. Syst.* 10.4 (2011), 46:1–46:24.

[5] Tommaso Cucinotta et al. "Multi-level Feedback Control for Quality of Service Management". In: *IEEE International Conference on Emerging Technologies and Factory Automation*. 2009, pp. 1–8.

[6] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. "Feedback Control for Adaptive Live Video Streaming". In: *ACM Conference on Multimedia Systems*. 2011.

[7] Viktor Edpalm et al. "Camera Networks Dimensioning and Scheduling with Quasi Worst-Case Transmission Time". In: *Euromicro Conference on Real-Time Systems (ECRTS)*. 2018, 17:1–17:22.

[8] Viktor Edpalm et al. *H.264 Video Frame Size Estimation*. Technical Report. Department of Automatic Control, Lund University, Sweden, Mar. 2018.

[9] Erwin Harahap et al. "A Model-Based Simulator for Content Delivery Network using SimEvents MATLAB-Simulink". In: *INSIST* 1.1 (2016).

[10] Joseph L. Hellerstein et al. *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.

[11] A. Leva and M. Maggio. "Feedback process scheduling with simple discrete-time control structures". In: *IET Control Theory Applications* 4.11 (2010), pp. 2331–2342.

[12] M. Maggio et al. "Controlling software applications via resource allocation within the heartbeats framework". In: *IEEE Conference on Decision and Control*. 2010, pp. 3736–3741.

[13] Alexandre Martins et al. "Control-Based Resource Management for Storage of Video Streams". In: *IFAC World Congress, Berlin, Germany*. 2020.

[14] Luigi Palopoli et al. "AQuoSA - adaptive quality of service architecture". In: *Softw., Pract. Exper.* 39.1 (2009), pp. 1–31.

[15] Luigi Palopoli et al. "Weighted feedback reclaiming for multimedia applications". In: *IEEE/ACM Workshop on Embedded Systems for Real-Time Multimedia*. 2008, pp. 121–126.

[16] R. P. Pothukuchi et al. "Control Systems for Computing Systems: Making computers efficient with modular, coordinated, and robust control". In: *IEEE Control Systems Magazine* 40.2 (2020).

[17] Iain E. Richardson. *The H.264 Advanced Video Compression Standard*. Wiley Publishing, 2010.

[18] Xiaoqi Yin et al. "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP". In: *ACM Conference on Special Interest Group on Data Communication*. 2015, pp. 325–338.