Resource Management in Distributed Camera Systems

Martins, Alexandre

2022

Document Version:
Publisher's PDF, also known as Version of record

Resource Management in Distributed Camera Systems

ALEXANDRE MARTINS
DEPARTMENT OF AUTOMATIC CONTROL | LUND UNIVERSITY

Printed by Media-Tryck, Lund 2022 NORDIC SWAN ECOLABEL 3041 0903

Department of Automatic Control
P.O. Box 118, 221 00 Lund, Sweden
www.control.lth.se

PhD Thesis TFRT-1135
ISBN 978-91-8039-188-7
ISSN 0280–5316

Resource Management in Distributed Camera Systems

Alexandre Martins

LUND
UNIVERSITY

Department of Automatic Control

PhD Thesis TFRT-1135
ISBN 978-91-8039-188-7 (print)
ISBN 978-91-8039-187-0 (web)
ISSN 0280–5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Media-Tryck.
Lund 2022

To my family and close friends who supported me during my ups and my downs, people at the automatic control department for being such an everyday inspiration and my personal raccoon for coping with me for all these years.

Thank you all for contributing to this day!

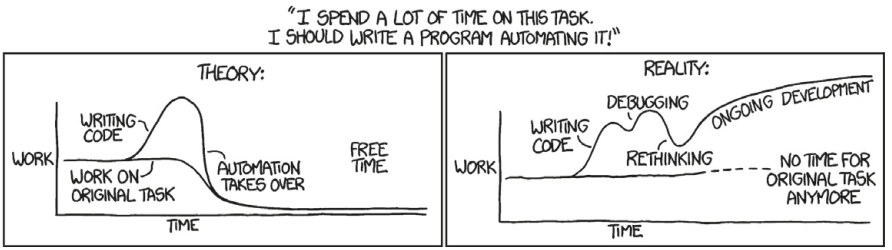
Abstract

The aim of this work is to investigate different methods to solve the problem of allocating the correct amount of resources (network bandwidth and storage space) to video camera systems. Here we explore the intersection between two research areas: automatic control and game theory. Camera systems are a good example of the emergence of the Internet of Things (IoT) and its impact on our daily lives and the environment. We aim to improve today's systems, shift from resources over-provisioning to allocate dynamically resources where they are needed the most. We optimize the storage and bandwidth allocation of camera systems to limit the impact on the environment as well as provide the best visual quality attainable with the resource limitations. This thesis is written as a collection of papers. It begins by introducing the problem with today's camera systems, and continues with background information about resource allocation, automatic control and game theory. The third chapter describes the models of the considered systems, their limitations and challenges. It then continues by providing more background on the automatic control and game theory techniques used in the proposed solutions. Finally, the proposed solutions are provided in five papers.

Paper I proposes an approach to estimate the amount of data needed by surveillance cameras given camera and scenario parameters. This model is used for calculating the quasi Worst-Case Transmission Times of videos over a network. Papers II and III apply control concepts to camera network storage and bandwidth assignment. They provide simple, yet elegant solutions to the allocation of these resources in distributed camera systems. Paper IV combines pricing theory with control techniques to force the video quality of camera systems to converge to a common value based solely on the compression parameter of the provided videos. Paper V uses the VCG auction mechanism to solve the storage space allocation problem in competitive camera systems. It allows for a better system-wide visual quality than a simple split allocation given the limited system knowledge, trust and resource constraints.

Preface

Being an engineer is a daily battle with the intricate balance between cost and results. With infinite amount of power, data, time, one could solve most things. But reality reminds you that everything has a cost. A good engineer should solve complex problems with elegant simple solutions. To be able to find good solutions, one needs to know the problem well. Around the beginning of my career I was assigned the task of creating a bit rate controller, a control loop which would influence the amount of data loss in videos in order to be able to deliver the video from cameras over the network to the recipient. Then I moved on to find ways to drop data in a smart way in order to limit the amount of resources used to produce the same images. It was not long before we realized that the problem did not reside in the camera and that thinking of the devices alone in the age of IoT was a lost battle. One needs to think of systems of devices and not of devices in a system. This is where the journey towards resource allocation began. Now that cameras could produce a defined amount of data and get rid of less useful data, how would they be able to give more room to others which need these resources right now? How can they be smarter devices and be part of a network of smart devices?



https://xkcd.com/1319/

Acknowledgements

Being part of the automatic control department has been inspirational and everyone made these years some of the most fun I had. Thank you all for being great people, especially to the amazing staff for keeping the spirit up!

Karl-Erik Årzén, I would not be here today without you. Thank you for being the best supervisor I could have wished for, supporting when needed, encouraging during inspirational times and still dragging back to earth when necessary.

Martina Maggio, Harry Pigot, Richard Pates, Johan Ruuskanen, Nils Vreman and Emil Vladu you are such amazing colleagues and friends and helped a lot during this work. Gautham Nayak Seetanadi, thank you for always being around, in research or in life, I am happy to count you among my friends. Per Skarin, you have been a great travel and office partner, I miss our discussions but look forward to new ones! Amir Roozbeh, Joris Van Rooij, Rebekka Wohlrab I look forward continuing our time together. Pr. Hung-Hu Wei, thank you for the warm welcome and guidance in my research.

My former colleagues from Axis Communications were also a big part of this work. Axis is full of great people who helped me move forward. I met amazing people and long-lasting friends there. Thank you Fredrik Pihl for being an great inspiration and amazing friend, Viktor Edpalm for all the great past, present (and future) good times, Mariana Rojas for your unending support and positiveness, Danilo Chinchilla Sosa for your wisdom and Yuan Song for always being such a great inspiration.

Mikael Lindberg, I will never be able to thank you enough for sponsoring me, being of great help in research and wine selection! Thank you for dragging me back to earth when I needed it while still looking at the sky.

Olivier Bonjour, you are my everyday support and confident. I am lucky to have your attentive and strong spirit around to challenge my daily craziness, your unending support fills both my spirit and belly with delicious cakes.

A warm thank you to my close family who always support me by all means possible. I would not be half of the person I am today without you all and you mean the world to me. Even if we are now spread over the world I hope to give back at least a tenth of what you gave me over the years.

Last but not least I would like to thank my friends for supporting me everyday over these research years, a big thank you to you Safina Khellaf, Flaurian Leroux, Guillaume Garnon, Jack Bates, Nina Latouille, Samia Mellouki, Luis Adrián Serratos Sotelo, Ludwig Rojas, Julien Schmitt and others.

And to all I have not mentioned in here, I could not have done this journey without many of you, because each action has a reaction, every interaction I had shaped what I am and do today... Thank you...

Financial Support

This work was supported by Axis Communications. It 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).

Contents

1

Introduction

Today's world is more and more interconnected. With the development of smart cities and the Internet of Things (IoT) trend, the amount of machine to machine communication is growing exponentially. With the advances in electronics, computer-like devices are present everywhere [Ahad et al., 2020]. A good example of this are camera systems, where nowadays each camera is a small specialized Linux computer with an image sensor running on an IP network. These new smart cameras run on-board image/video analytics, have machine learning acceleration hardware to, e.g., do object detection and tracking, image optimization or to self-reconfigure [Bellman et al., 2020]. They run continuously, analyzing the images provided by sensors, running algorithms in quasi real-time and sending information and video data to other machines over the network.

The functioning, transmitting, processing and storage of information produced by the different devices consume resources.

The emergence of these connected devices allows for smarter cities and the promise of more efficient, innovative, greener and more durable systems [Kitchin, 2015]. Unfortunately most of these systems are rarely optimal and resources are often wasted due to poor provisioning. Most systems allocate resources by coarsely estimating the amount of resource needed in the worst case scenario, and then they over-provision to allow for these potential extreme scenarios. This approach leads to a lot of resource waste, which by extension leads to a cost for society as well as a greater impact on the environment, contradicting the promise of greener automated cities.

By estimating resource usage in a better way and introducing dynamic adjustment to the resource allocation scheme we can reduce the impact of such systems. This will ensure a better overall quality of service while keeping the costs of running it as low as possible and without waste of energy, money or data.

Optimizing the resources needed to achieve a certain task can be thought of in two ways, focusing either on Quality of Service (QoS) or Quality of Experience (QoE). The QoS and QoE are defined by the use case of the

system. QoS is more focused on the availability of services while QoE focuses more on the quality from the point of view of the final user of the system [Chen et al., 2014]. In most of our work the Quality of Experience (QoE) is seen as the video quality the viewer receives from the different cameras. The first way to approach the problem of allocating resources is to maximize the QoE based on resource availability, i.e., get the most out of the system given known or measured limitations. The second way, on the contrary, is to minimize the usage of resources in order to achieved a certain acceptable QoE level. The first approach is more focusing on optimal allocation with the provided constraints while the second focuses more on resource saving.

The number and size of camera systems used, e.g., in different types of public spaces with surveillance cameras, are growing and they are currently one of the major consumer of storage and bandwidth. With growing demands on high resolution, high frame rate and level of detail, the amount of storage needed to retain these videos is a growing problem (as explained in [IPVM, 2021]). Surveillance camera systems are usually critical installations and are thus mostly running on dedicated infrastructures, storing video in trusted servers owned by system administrators. Newer installations are usually large scale (commonly hundreds of cameras), heterogeneous and have large differences in resource requirement. The amount of bandwidth or storage available is usually limited due to legacy or cost constraints. This is where optimizing camera systems becomes relevant attempting to obtain the best image quality given the limited and variable amount of resource available, regardless of the type and size of the system.

In this thesis we focus on two resources: storage and network bandwidth. These resources are dependent, i.e., the allocation of each of these influences the allocation of the other, e.g., one needs to have enough network resources in order to be able to store a certain amount of data on the receiver end. It is pointless to allocate a lot of storage space if the amount of bandwidth allocated is insufficient, and vice versa. One important point is that even if dependent, those two resources usually do not change at the same pace. Usually the variation of the global amount of bandwidth available changes faster than the storage amount available.

Here we try to bridge two classical techniques used for resource allocation: game theory and automatic control. Game theory has historically been used in bandwidth assignment in mobile telecommunication networks due to their scale and the lack of trust between the devices, where mini-auctions are performed to split the bandwidth between devices at mobile base stations. Automatic control on the other hand has been used for more specific and time critical applications such as memory bandwidth allocation or coordination of distributed systems.

1.1 Thesis Outline

This thesis is written as a collection of papers, with the following outline:

Chapter 1 - Introduction

The introductory chapter describes the motivation and aim of the thesis. It introduces the main contributions and the included papers.

Chapter 2 - Resource Management and Auction Theory

The second chapter presents important concepts required to understand the papers. It starts by explaining the resource management concept and methods used in resource allocation and continues by explaining auction and game theory, its history and main principles.

Chapter 3 - System Models

This chapter presents the camera system models considered in the papers and exposes their constraints and specificities. It has been divided into control and pricing constraints as each approach presents slightly different limitations. In this chapter we also give an overview of video compression and video quality estimation techniques which are central in the model described in Paper I and then used in following Papers II-V.

Chapter 4 - Control Background

This chapter briefly explains the automatic control techniques used in Papers II to IV. We start by explaining PID control and the related tracking-based anti-windup and continue with cascade control where multiple controllers are cascaded to control a process. We then introduce mid-ranging where two controllers are combined to control a process with two actuators, and finish by explaining how bumpless mode changes between controllers can be done.

Chapter 5 - Game Theory Background

This chapter briefly explains the pricing and game theory techniques used in Papers IV and V. We start with pricing theory by explaining the price discrimination used in Paper IV. We then continue with the introduction of the Vickrey-Clarke-Groves auction mechanism used in Paper V and its related knapsack problem which is used for the allocation decision.

Chapter 6 - Conclusion and Future work

The final chapter provides a summary of the work done, its results and gives suggestions for future improvements.

Contributions of the Thesis

The thesis is based on five publications. Paper I proposes an approach to estimate the amount of data needed by surveillance cameras given camera and scenario parameters. The proposed model has been compared against state of the art methods and provides a better estimate of the video frame sizes. Papers II and III apply control concepts to camera network storage and bandwidth assignment. They provide simple, yet elegant solutions to the allocation of these resources in distributed camera systems. Papers IV and V propose a valuation model of the amount of storage based on the model proposed in Paper I. This valuation is used to solve a similar resource allocation problem to that in Papers II and III. Paper IV combines pricing theory with control techniques to force the video quality of camera systems to converge to a common value based solely on the compression parameter of the provided videos. Paper V uses the VCG auction mechanism to solve the storage space allocation problem in competitive camera systems. It allows for a better system-wide visual quality than a simple split allocation given the limited system knowledge, trust and resource constraints.

Paper I

Edpalm, V., A. Martins, K.-E. Årzén, and M. Maggio (2018). "**Camera Networks Dimensioning and Scheduling with Quasi Worst-Case Transmission Time**". In: Altmeyer, S. (Ed.). *30th Euromicro Conference on Real-Time Systems (ECRTS 2018)*. Vol. 106. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Barcelona, Spain, 2018, 17:1–17:22.

In this paper we propose a set of simple measurable parameters which when combined allow to predict the expected H.264 frame sizes and by extension the video bandwidth of each device. It proposes a method to tailor the storage and/or network bandwidth required for a surveillance system in advance using information about which cameras will be used and under which conditions. The paper describes a method to compute frame size estimates to be used in quasi Worst-Case Transmission Time (qWCTT) calculations for cameras that transmit frames over IP-based communication networks. The precise determination of the qWCTT allows us to model the network access scheduling problem as a multiframe scheduling problem and to re-use theoretical results for network scheduling. The paper presents a set of experiments, conducted in an industrial testbed, that validate the qWCTT estimation. The paper is the product of years of work in a surveillance camera company, gathering content specific to its use case and allowing video engineers to craft a rough model of the video bandwidth. It is based on an extensive work done by V. Edpalm and

A. Martins in the gathering of data, model development, validation against state of the art techniques and compilation of results. M. Maggio helped drawing a parallel between network scheduling for video surveillance camera systems and CPU scheduling. The work was reviewed and validated by K.-E. Årzén. This work has been extended in a technical report, which provides more details and tests, [Edpalm et al., 2018].

Paper II

Martins, A., M. Lindberg, M. Maggio, and K.-E. Årzén (2020). "**Control-Based Resource Management for Storage of Video Streams**". In: vol. 53. 2. 21st IFAC World Congress. Berlin, Germany, 2020, pp. 5542–5549.

In this paper the basic concept and limitations of the systems considered for the thesis are described and a control-based approach to solve it is proposed. 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. The disk space allocation is formulated here 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 based on a simple linear model as well as the model described in Paper I. The initial idea for this approach was proposed by K.-E. Årzén and then extended by A. Martins. Model development, simulation, data gathering and writing was done by A. Martins with input from K.-E. Årzén. The work was finally reviewed by M. Maggio and M. Lindberg.

Paper III

Martins, A. and K.-E. Årzén (2021). "**Dynamic Management of Multiple Resources in Camera Surveillance Systems**". In: *2021 American Control Conference (ACC)*. New Orleans, United States, 2021, pp. 2061–2068.

This paper is a logical extension of the work in Paper II. Here, the disk space allocation problem and the network bandwidth allocation problem are solved jointly using techniques normally associated with process control such as mid-range control and the tracking-based control of global shared resources proposed in Paper II. The approach helps to address the allocation of limited dependent resources, i.e., 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). It shows how known control techniques can be combined to solve the allocation of joint resources. This work was done and written by A. Martins with valuable input and review from K.-E. Årzén.

Paper IV

Martins, A., H.-Y. Wei, and K.-E. Årzén (2022). "**Storage Allocation for Distributed Video Sensor Systems using Feedback-Based Price Discrimination**". In: *Proceedings of the 11th International Conference on Sensor Networks - SENSORNETS,* SciTePress, Virtual conference, pp. 34–44.

This paper bridges control theory and pricing theory. We propose a framework using feedback-based price discrimination of storage resources in order to guarantee a uniform quality level of the videos in camera sensor networks, regardless of the specific camera sensor parameters. A lightweight solution is used that combines price discrimination principles from micro-economics, simple video quality metrics, cascade control, PI (Proportional and Integral) controllers, and a probing controller to allocate storage resources, while separating the resource providers (i.e., the storage units) from the resource buyers (i.e., the camera sensors). The buyers have private information on the amount of resources needed and act accordingly to maximize their utility (here the desire to minimize the compression of their own video stream). The storage units enforce the constraint on resource availability and fairness through the use of pricing. This work was developed, simulated and written by A. Martins with input and from K.-E. Årzén and reviewed by H.-Y. Wei (National Taiwan University, Taiwan).

Paper V

Martins, A., H.-Y. Wei, and K.-E. Årzén (2022). "**Vickrey-Clarke-Groves Auction-Based Storage Allocation for Distributed Camera Systems**". In: *Under conference submission*.

In this paper we address the resource storage allocation problem using techniques from auction theory. We designed an auction framework based on the Vickrey-Clarke-Groves (VCG) auction mechanism in order to allocate the available storage resources optimally. Each device is provided a fixed budged and tries to maximize its own private utility (based on the money spent, compression amount and its variation over time) in a competitive system

with constrained resources. The advantage of VCG auctions is that they provide guarantees, such as a fair and envy-free allocation without requiring control over all devices participating in the auction. The work was developed, simulated, tested and written by A. Martins with valuable input from H.-Y. Wei (National Taiwan University, Taiwan), it was reviewed and supported by K.-E. Årzén.

2

Resource Management and Auction Theory

In this chapter, we provide background on the theory and techniques used in the papers. Starting with the basis of resource allocation, we continue by describing some of the available resource allocation methods. We then focus on game and auction theory and its main principles.

2.1 Resource management

A system resource is any physical or virtual component needed by a computer system. It has usually limited availability. All connected devices and internal system components are resources, e.g., CPU cycles, memory, network bandwidth, storage space, electricity, etc. Resource management refers to the techniques for managing these limited resources. Its goal is to allocate the (sometimes) scarce resources to the correct users or clients at the correct time, while preventing resource leaks (allocating resources which are not needed anymore) and dealing with resource contention when the same resources are needed by multiple users that need to be prioritized. Networking, computing and energy resources are allocated taking into account hardware, performance, energy and environmental restrictions.

Most of the resources allocated to some extent depend on the allocation of other resources. A typical example is CPU time and memory. If a software component needs to run algorithms, it usually needs both computing power and storage space (to store the information needed for the computation). Lacking one of those would prevent the software from running and allocating too much of it would be wasting resources which could be used by other software components. Another example used in this work is network bandwidth and video storage space. There is no point in allocating a lot of storage space but no network resources because the data then could not reach the destination.

Resource allocation methods

Shared resource allocation is crucial is Real-Time Operating Systems (RTOS). In RTOS multiple tasks (or processes or threads) with more or less time-critical deadlines are sharing one or several CPUs. The tasks often need to access external devices, e.g., the screen or the keyboard, and execute code sections under mutual exclusion constraints, i.e., only one task (or a limited number) may execute inside the section at the same time. Code sections of this type go under the name critical sections and use mutual exclusion to prevent simultaneous access to a shared resource. To obtain the necessary mutual exclusion synchronization a number of mechanisms have been developed, e.g., locks, semaphores, and monitors, [Sha et al., 1990]. Synchronization protocols such as priority inheritance or priority ceiling protocols [Rajkumar et al., 1988] are then used to minimize the blocking times and/or to avoid deadlocks (a state where two or more processes hold a resource which prevents other processes from proceeding).

Some resources are divisible and others are not. External devices and critical code sections are examples of resources that are indivisible, i.e., a user either gets access to the entire resource, either alone or with a limited number of other simultaneous users, or gets no access at all. In this thesis we focus on resources that are divisible, i.e., which can be shared between multiple users so that each resource gets a certain share or budget of the resource. Some resources, e.g., memory, are divisible in themselves, whereas other resources can be made divisible using scheduling. One example of the latter is CPUs that can be made divisible using reservation-based scheduling, e.g., [Shin and Chang, 1995]. In this case the scheduling mechanism switches or multiplexes between the users in such a way that each user gets a certain share or budget of the resource (called a reservation) over a certain time interval. This can be done for example with time-slicing (giving access to the resource for a certain time window), space or frequency division (providing different paths/frequencies to different processes), or angular/polarization division (where the resource is split into different polarizations) of the particular resource. A good overview of reservation-based scheduling can be found in [Lindberg, 2007].

When allocating divisible resources there is always a global constraint that must be fulfilled, i.e., that the sum of the budgets of all the users never exceeds the total available amount of resources, i.e., 100%. In this thesis we focus on two divisible and dependent resources: storage and bandwidth.

Resource allocation approaches

Resource allocation can be achieved in different ways. Three of the main approaches are optimization-based allocation, control, or feedback, based allocation, and game theory-based allocation. These approaches can also be

combined.

Optimization based resource allocation: In this approach the problem is formulated as an optimization problem. The resource amounts are the decision variables and the optimization objective could be to either to maximize the performance subject to some resource constraints or minimize the resource consumption subject to some performance constraints. An example of the first situation can be seen in Eq.2.1.

$$\begin{aligned} &\text{maximize} && \textstyle\sum_x f(\text{resource}_x) \\ &\text{subject to} && \textstyle\sum_x \text{resource}_x \leq \text{Total resource availability} \end{aligned} \tag{2.1}$$

where $f()$ represents the performance provided by the allocated resources and x is the the resource consumer index.

The type of application and resources decide what type of optimization problem that is required, e.g., integer programming (ILP), mixed-integer linear programming (MILP), quadratic programming (QP), or nonlinear programming (NLP) methods such as branch & bound, genetic algorithms, constraint programming, or knapsack. Solving an optimization problem is often relatively time-consuming so therefore this type of resource allocation is preferably done either off-line or on-line when a major change has occurred.

Control-based resource allocation: In control-based resource allocation one uses feedback control to keep some performance or quality variable at a desired target or reference value, alternatively to maximize or minimize some performance or cost function, using the resource budget as the control signal. If we use single-input single-output (SISO) control then the resource budget for each individual user is determined by a separate controller (as illustrated by Fig. 2.1). Hence, some coordination mechanism is needed between the controllers to enforce the global resource constraint. This is the topic of Papers II and III. If one uses a multiple-input multiple-output (MIMO) controller that supports constraint handling then the resource constraint could be included in the controller. One example of this is Model Predictive Control (MPC), [Camacho and Alba, 2013], where the control signals (resource budgets) are computed online by solving an optimization problem each sample. This, however, normally leads to a centralized architecture which is not always desirable. Therefore, this has not been studied in this thesis. MPC is also an example of how optimization and feedback-based resource allocation can be combined.

Game/auction theory-based approaches: Game theory models situations in which decision makers have to make specific actions that have mutual

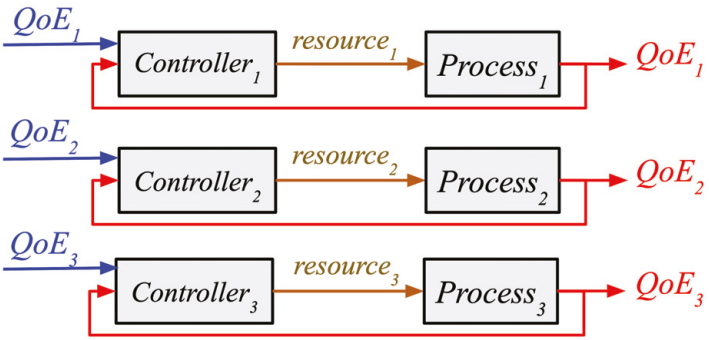


Figure 2.1 A simplified example of control-based resource allocation. The blue QoE represents the desired quality and the red QoE represents the actual quality.

consequences. This method is often used in distributed systems to decentralise the decision process and allocate resources in systems with multiple (often untrustworthy) actors. It is used to decide which process or actor should get the resources based on the valuation the process (or actor) has of the resources. The basics of game and auction theory are explained in Section 2.2. Auction theory has been used for many resource allocation problems such as mobile broadband spectrum allocation [Milgrom, 2004], virtual server resource allocation [Wei et al., 2016], device-to-device radio resource allocation [Pang et al., 2014], mobile network video caching [Li et al., 2016], camera network area coverage based on task allocation [Rinner et al., 2012; Lewis et al., 2013], etc. A nice overview of other game theory applications to resource allocation in computer systems is available in [Charilas and Panagopoulos, 2010] and a more detailed walk-through can be found in [Niyato et al., 2020].

2.2 Auction theory

Auctions facilitate transactions by enforcing a specific set of rules regarding the resource allocations to a group of bidders. Auction theory originates from game theory and economics theory. It deals with how bidders act and react in auction markets and investigates how the features of auctions provide incentives towards predictable outcomes. It is a tool used in the design of real-world auctions such as selling auctions, voting, market determination, etc. Auction theorists design rules for auctions to address issues which can lead to market failure. Usually, the auctioneer determines who is awarded the item(s) based on the agents' bids. The objective of the system designer is to engineer the costs, revenue and auction mechanism in such a way that individual self-interest leads to globally efficient solutions.

Some history

The history of auctions extends back to 500 B.C. in Babylon when Herodotus reported the use of an auction to sell women on the condition of marriage. Within the Roman Empire the "atrium auctionarium" (seamlessly increasing open-cry auctions) was used by soldiers to sell goods acquired "sub hasia" (under the spear). The global slave trade was also conducted in major port cities that imported slaves from around the world. These slaves were sold at auctions both to wholesalers and to individuals. Soon after the French Revolution auctions were conducted in taverns and coffeehouses to sell art. Such auctions were held daily, and catalogs were printed to announce available items. Large auction houses such as Christie's and Sotheby's arose in the mid-1700s and provided an organized forum for the buying and selling of disparate goods at auctions.

Auction theory itself begins to be formalized in the 1990s when John Nash (1994 Nobel Laureate) designed a generalized theory of auctions as a non-cooperative game which moves beyond simple zero-sum games. Vickrey (1996 Nobel Laureate) and Harsanyi (1994 Nobel Laureate) extended Nash's equilibrium specifying ways in which equilibria can be reached under an information set (specific set of shared and private information between players). By the end of the 1990s, auction theorists had defined equilibrium bidding conditions for single-object auctions under most realistic auction formats and information settings. In 2020, Robert B. Wilson and Paul Milgrom won The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel for their work on defining how multiple-object auctions can be performed efficiently.

Main principles

A game-theoretic auction model is a mathematical game represented by a set of players, a set of actions (strategies) available to each player, and a payoff vector corresponding to each combination of strategies. Each bid function maps the player's value (in the case of a buyer) or cost (in the case of a seller) to a bid price [Krishna, 2009]. The payoff of each player under a combination of strategies is the expected utility (or expected profit) of that player under that combination of strategies.

Auctions can be centralized (one auctioneer handles the auction), distributed (auctioneers seek bidders and deal directly) or peer-to-peer (auctioneers and bidders cooperate and a reward is needed to ensure cooperation). Auctions can be online of offline. Online auctions are triggered on resource requests, each auctioneer has its own auction of items and the market is cleared immediately. In offline auctions the auctioneers provide time slots, wait for bids from all potential bidders or the end of a clearance period, and select how to allocate the item(s) [Krishna, 2009].

There are two main categories of auctions possible for a single item: open-cry and sealed-bid. Open-cry are auctions where all bidders see other bidders proposals (bids) while in sealed-bid auctions bidders provide their bids only to the auctioneer and others can only guess the bids of others.

The main open-cry auctions (everyone know the others bids) are:

- Open ascending-bid auctions (English auctions): Each bidder makes increasingly higher bids. This continues until no participant is making a higher bid; the highest bidder wins the auction at the final amount of the bid [Krishna, 2009].

- Open ascending-bid auctions (Japanese auction): This is a variant of the English auction where the amount of the bid is set with an increasing price clock. All bidders join a bidding area and the clock starts at the lowest valuation of the item. Bidders who reached their maximum bid leave the bidding area. The auction stops when only one bidder is left in the bidding area, the amount due is the one set by the clock [Krishna, 2009].

- Open descending-bid auctions (Dutch auctions): The price of the item is set by the auctioneer at a high level. The amount is then progressively lowered until a bidder is prepared to buy at the current price, winning the auction [Krishna, 2009].

- Anglo-Dutch auctions: The price increases and buyers are dropped, when only two are left they start a first-price closed auction [Krishna, 2009].

- Waiting-line auctions: Buyers join a queue, the items are allocated in a first come first serve manner. Being first in line means having paid more to get the item [Taylor et al., 2003].

Sealed-bid auctions (all bidders provide their secret bid to the auctioneer) can be classified as:

- First-price auctions: The bidders place their hidden bid (in a sealed envelope for example) and simultaneously hand them to the auctioneer. The bidder with the highest bid wins, paying the amount of the proposed bid for the item [Krishna, 2009].

- Second-price auctions (Vickrey auctions): The bidders again place their hidden bid and simultaneously hand them to the auctioneer. The bidder with the highest bid wins, paying a price equal to the second-highest bid [Vickrey, 1961].

- K'th-price auctions: This is an expansion of the Vickrey auction. The bidder with the highest price wins but pays the k'th highest price [Krishna, 2009].
- All-pay auctions: Every bidder pay regardless of whether they win the item or not, the item is awarded to the highest bidder. A typical example are Tullock auctions or lotteries [Taylor et al., 2003].

In addition to the above auction types, auctions can also present different properties. They can be one shot auctions where only one bid or set of bids is provided for an item or multi-shot where multiple bids are done sequentially, each bidder sends its one-shot bid for a single item, then another for the second item, etc. One shot bids can be done for a single item (bidders only compete for a single item from the auctioneer and each item is evaluated separately), multiple items (bidders send a list of multiple item/price pairs to the auctioneer). The bidders can provide single pairs of item/value to the auctioneer or send a price-demand function, i.e., a piece-wise linear price/demand function reflecting its bidding strategy, or the value function can be pre-determined and the bidders just provide an index for a pre-determined function (provided by the auctioneer or shared) [Krishna, 2009].

In this thesis we allocate multiple identical items. For this we use the Vickrey-Clarke-Groves auction (VCG), i.e., use a centralized, offline, multiple item, second price sealed-bid auction where the auctioneer maximizes the total revenue from selling the items [Nisan and Ronen, 2007].

Auctions can be seen as games where each bidder and auctioneer is a player. As such, some important game properties are important to know [Owen, 2013]:

- Cooperative / non-cooperative game: A game is cooperative if the players are able to form binding commitments and it is non-cooperative if players cannot form agreements apart from the ones enforced by the game.
- Symmetric / asymmetric game: A symmetric game is a game where the payoffs for playing a particular strategy depend only on the other strategies employed, not on who is playing them.
- Zero-sum / non-zero-sum game: Zero-sum games are constant-sum games in which choices by players can neither increase nor decrease the available resources. In zero-sum games, the total benefit to all the players in a game always sums to zero.
- Simultaneous / sequential games: Simultaneous games are games where both players move simultaneously or where the later players are unaware of the earlier players' actions. Sequential games are games where later players have some knowledge about earlier actions.

Each game can have some optimal outcomes which are enforced by the rules of the game/auction. Keep in mind that it could be that there is no preferred outcome depending on the rules and players strategy. Some of these outcomes are:

- Dominant strategy: A strategy is dominant when the strategy is better than another strategy for one player, no matter how that player's opponents may play [Krishna, 2009].
- Nash equilibrium strategy: A solution where no player has anything to gain by changing their own strategy. A Nash equilibrium is a dominant strategy. A game may have multiple Nash equilibria or none at all. The Nash equilibrium does not always mean that the most optimal strategy is chosen, it just indicates that each player's strategy is optimal when considering the decisions of other players. Every player wins because everyone gets the outcome they desire [Osborne and Rubinstein, 1994].
- Pareto optimal strategy: A solution where no player can be better off without making at least one other player worse off or without any loss thereof. Pareto efficiency implies that resources are allocated in the most economically efficient manner, but does not imply equality or fairness [Krishna, 2009].

3

System Models

A typical video surveillance system consists of multiple cameras disseminated over an area. These cameras continuously record a specific scene, for instance an office space, a parking lot, a road, or any other alternative. The recorded scenes are different from one another, but their characteristics do not usually change significantly over time. A camera that is installed outdoor in a parking lot will record similar scenes, mostly involving cars and people, in different light conditions. At the same time, a camera that is pointing to a highway lane will (most likely) record either an empty road, or the passage of cars.

A common challenge in the video surveillance industry is to tailor the entire infrastructure of the surveillance system to achieve a certain level of quality, while keeping the cost as limited as possible. The main challenge is that these systems become more and more heterogeneous, complex and interconnected. Previously only large institutions were equipped with such surveillance systems and had dedicated infrastructure (network and storage units) to handle the expected load of information produced. However, today's systems are usually sharing the same network infrastructure as other network users and devices, even sometimes running over the cloud or on the internet with disseminated recording installations that could be on another continent. The legacy solution of allocating a fixed amount of resource to each device does not hold anymore and new automated and distributed techniques must be explored.

In this chapter we will define the models and constraints considered to the describe the camera systems that are the topic of the final five publications.

3.1 Camera Systems

Today, the video industry is mainly focused on using IP cameras, which stream videos that are compressed using the H.264 standard [ITU-T, 2010]. These cameras are connected through an IP network which can be shared with others or dedicated. Each camera streams video to one or multiple storage

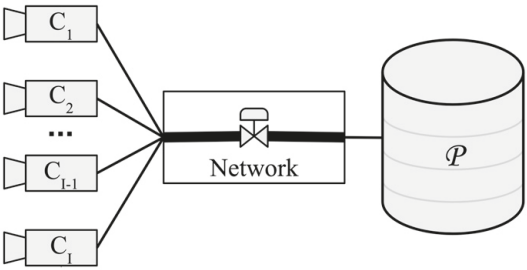


Figure 3.1 The simplified system considered in this thesis.

locations which can be local, remote or a combination of both. A typical camera network has at least one storage unit $\mathcal{P}$ (Network Attached Storage, Cloud storage or other) and I video cameras, C, indexed with i: $\{C_1, C_2, ...C_I\}$ (see Fig. 3.1).

Typically a video surveillance system is owned by a security department, which buys or rents storage from an IT department or a cloud provider at a fixed rate. In our systems, viewing quality is considered as most important. The main system goal is to maximize the overall global video quality given the current system constraints, i.e., the running cost and the video storage size.

It is assumed that all cameras in the network can at least communicate with the storage unit. The total quantity of storage available by the storage provider is s and the storage space allocated to camera C_i is s_i. The cameras do not have internal storage space, i.e., only the storage unit has storage space available to store videos.

The total amount of storage available is fixed and allocated during system design. The amount of storage to be used over a certain period is defined by the system owner based on design choices or regulations. A typical example is that video surveillance footage should be saved for 30 days and destroyed afterwards. The surveillance system owner then needs to try to allocate the total amount of storage s to accommodate the needs of the cameras.

Each camera C in the system is independent and does not communicate with other cameras. Its only purpose is to continuously send the video it records to a storage location. Each camera has different internal characteristics (sensor type, encoder properties, lens type, video resolution, frame rate, etc.) and is filming different environments with unique parameters (light condition, amount of motion, etc.). This combination of parameters results in large differences in the amount of data each camera is producing. The amount of data produced by a camera C is usually measured in kilobits per second (kb/s) or megabits per second (Mb/s). More details about how much each camera is estimated to produce is given in Section 3.2.

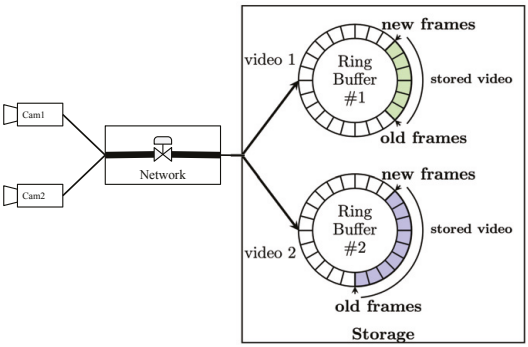


Figure 3.2 The control setup considered in this thesis.

Control system constraints

In the first two papers in the thesis we are given a fixed amount of available global storage, i.e., disk space and we want to store 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 the video it produces 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 to adjust their video compression also based on the global constraint. We consider the storage provider to act like a set of ring buffers, containing a sliding window of stored frames for each camera. When new data is coming in from a camera and the buffer is full, old data is deleted in order to accommodate this. The time video frames stay in the buffer represents the retention time of the stored videos. The amount of storage allocated to each camera is the size of the ring buffer. An illustration of the system can be seen in Fig. 3.2.

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 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.

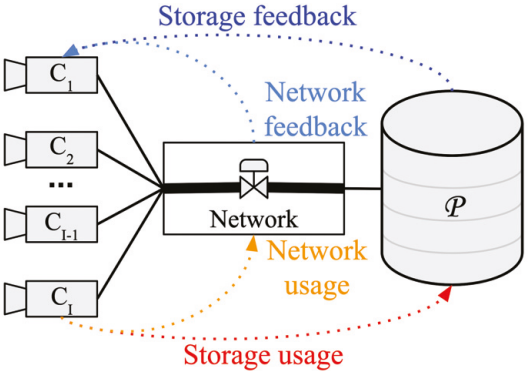


Figure 3.3 The simplified system considered in the control papers.

Too high bandwidth constraints on the network infrastructure will delay or possibly cause data packet 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.

The model used in this approach consists of two to three parts: the camera model, network model, and storage model. The camera model consists of the encoder which calculates the bandwidth of the video stream as a function of the compression level. 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 budget, of the total network bandwidth. Finally, the storage model emulates the ring buffer behavior of the storage provider. An illustration of the system can be seen in Fig. 3.3.

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.

Pricing/auction constraints

In the second part of the thesis, we move from a cooperative system, where each camera is expected to react to the provided global feedback regarding resource availability, to a competitive system where each camera would try to maximize its own video quality regardless of the other resource consumers. The resource allocation can be seen as a zero-sum game where the total amount of storage available is fixed and a camera getting more resource

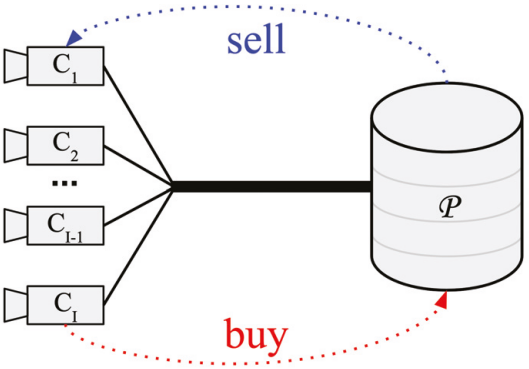


Figure 3.4 The simplified system considered in the pricing/auction papers.

means that another would get less resources.

For this we design a framework where, at every predefined period k, e.g., an hour, a day, or a week, the cameras need to obtain storage by purchasing it from the seller to save the video they generate, using virtual money.

At each period, k, the cameras obtain an amount of money, m, that they can use at their discretion to buy resources. The amount they receive depends on the cost of running the system. Each camera has a virtual account holding the money it may use. Any remaining money can be saved for future periods. An illustration can be seen in Fig. 3.4.

In paper IV, we use the price discrimination principle from microeconomics to allocate storage resources. The storage units enforce the constraint on resource availability through the use of pricing. The goal of the storage provider is to converge to an optimal and uniform quality level for all cameras in the system given the global resource constraints. In this approach, each camera is provided a unit cost for the resources which is determined centrally. Since the amount of money available is limited this will incite the cameras to adjust the amount of data they produce, and by extension, the visual quality level of the video, to a global value.

In paper V, we use auction theory, in the form of the Vickrey-Clarke-Groves (VCG) auction mechanism. Each camera still periodically gets the same amount of money and provides the storage provider with a set of bids containing the amount of resource it desires to acquire and the associated cost it is willing to pay for it. The storage units enforce the constraint on resource availability by solving a constrained knapsack problem to allocate the resources.

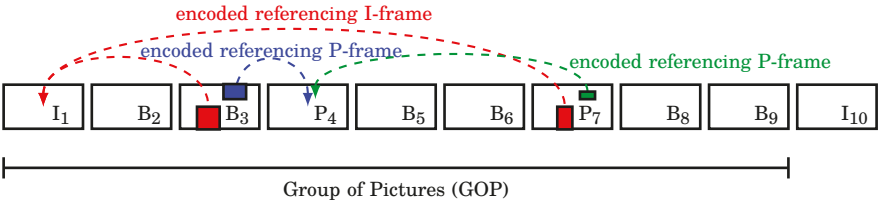


Figure 3.5 H.264 frame sequence: I-frames, P-frames, B-frames, and Group of Pictures.

3.2 Video compression

In order to tailor the infrastructure, one must be able to anticipate how much data each camera in the system is expected to produce given its unique set of internal characteristics and settings, e.g., position, placement, surrounding environment, etc. This is the topic of Paper I. The model presented in the paper is inspired by the way the H.264 standard compresses videos and how the environment influences such algorithms.

H.264 video encoding

H.264, also called MPEG-4 part 10 AVC, is a video compression standard that defines how a video should be decoded. The implementation of the encoding is left to the manufacturer's discretion [ISO/IEC MPEG & ITU-T VCEG, 2003]. The standard describes a *block based hybrid codec*, i.e., a video is decomposed into blocks of data for encoding. To allow for video compression, H.264 uses motion-compensated encoding, i.e., it describes a frame by referencing parts of other frames, thus capturing the motion of objects across different frames. A stream encoded with H.264 contains a sequence of frames, which are not necessarily encoded following the display order or the time when they were captured. Based on the frame encoding, it is possible to distinguish between three different types of frames: Intra frames (I-frames), Predicted frames (P-frames), and Bi-directional predicted frames (B-frames) [Wiegand et al., 2003].

Intra frames are self contained, i.e., parts of the image can only be predicted from other parts of the same image which requires a large amount of data. Predicted frames can refer to previous frames for parts which were already transmitted, which leads to savings on the amount of data needed to encode these frames as most has already been transmitted. Bi-directional predicted frames are the same as Predicted frames but they can refer to both previous and future frames.

The frames are organized in Group of Pictures (GOP) which consist of an I-frame followed by a sequence of B-frames and P-frames. Fig. 3.5 shows

a sequence of 10 frames (or images). The first nine frames in the sequence denote a Group of Pictures (GOP). The I-frame can be marked as an Instantaneous Decoding Refresh (IDR), meaning that the following frames do not need information from frames prior to that one in the sequence. If all the I-frames are marked as IDR points, the decoding of each GOP is independent, otherwise it is not.

For the sequence shown in Fig. 3.5, the first and the last frame are encoded as I-frames. The fourth and the seventh are encoded as P-frames. The remaining ones are encoded as B-frames. The red arrows in the Figure indicate areas of the third and seventh frames – respectively a B-frame and a P-frame – that are encoded as references to the previous I-frame. The blue arrow shows an area of the third B-frame that is encoded as reference to the following P-frame. The green arrow shows an area of the seventh P-frame that is encoded as a reference to the previous P-frame.

The given frames are split into *macroblocks*. To be more precise, a generic H.264 frame is split into multiple 16×16 squares of pixels, each of them being a macroblock. Macroblocks are encoded/decoded separately from one another, and can be split into sub-blocks down to a block size of 4×4 pixels [Wiegand et al., 2003]. Macroblocks are also assigned a type from the set {I, P, B}. I frames can contain only I-blocks. P-frames can contain both P-block and I-blocks. B-frames can contain all types of blocks.

Fig. 3.6 shows an overview of the encoding process. The input frame is divided into macroblocks, each of them is passed to a Coder Control and to a Motion Estimation function. The Motion Estimation function uses some previously encoded and buffered frames, the number of them being determined by the Coder Control. These previous frames are used to choose if the current block should be encoded:

- as a new block, containing the full information (Intra-Frame Prediction, I-block),
- by referring to a previously encoded block in the same frame, containing a positional vector and the residual information (Intra-Frame Prediction, I-block),
- by referring to a block in a previous frame, containing a positional vector, the frame reference, and the residual information (Motion Compensation, P-block), or
- by referring to block in a previous or future frame, containing a positional vector, the frame reference, and the residual information (Motion Compensation, B-block).

The Motion Estimation function determines the cost for the four choices and selects the most appropriate one for the current macroblock. The residual

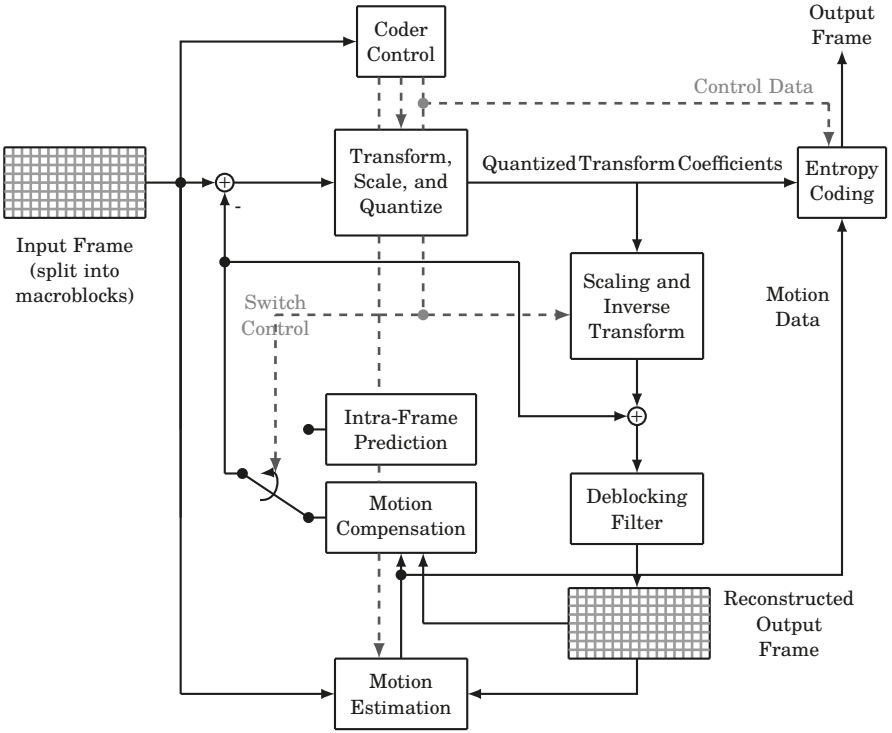


Figure 3.6 Basic coding structure of a H.264 frame.

information is then Transformed, Scaled, and Quantized according to a quantization parameter (QP) to reduce its size. Note that the quantization step is the *only* step in the encoding process where there is information loss. The higher the QP value, the higher the loss of information is in the image.

The quantization parameter is an integer value between 1 and 51 which indicates how much of the residual information should be kept. The scaling, inverse transform and the deblocking filter allow the encoder to reconstruct the output frame and buffer it for future encoding. The entropy coding function uses lossless statistical compression to produce the final output frame [Wiegand et al., 2003].

Video quality evaluation

In this thesis, we focus on the resource allocation problem and use the crude but simple visual quality parameter available from the video compression algorithm, i.e., the quantization parameter (qp) value.

Human perception of video/image quality does not correlate directly and

solely with the compression level qp and is influenced by cultural and social trends. Nevertheless the quantization parameter of H.264 videos, and its variation over time have a direct impact on the perceived video quality (using mean opinion score testing) according to [Xue et al., 2010; Xue et al., 2013; Lin et al., 2012], i.e., the lower and less varying the quantization parameter is, the better the perceived quality will be.

Image quality metrics is an active research topic which involves many adjacent areas such as biology and machine learning, as modeling human visual perception is a very complex topic. Despite the limitations of the qp metric, we argue that this is a valid first step. A logical future work would be to replace the quality parameter with another quality metric from the literature. We briefly present some quality metrics below.

We can isolate two types of video quality metrics [Brunnström et al., 2021]:

- subjective, that present the images to groups of people and ask them to grade their view of the quality (such as mean opinion square).
- objective, that use image analysis to extract a value or set of values which can be compared.

Objective metrics can be split into three method groups [Brunnström et al., 2021]:

- Full reference method, where one is assumed to have access to the "ground truth" image that is not altered. The most used methods are peak signal-to-noise ratio (PSNR) [Teo and Heeger, 1994], structural similarity index measure (SSIM) [Wang et al., 2004], video quality metric (VQM) [ITU-T, 2004] or video multi-method assessment fusion (VMAF) [Li et al., 2016].
- Reduced reference method, where one has access to information (reduced image, features information, etc) about the "ground truth" image [Dost et al., 2022].
- No reference method,where one has no access to the original image [Shahid et al., 2014].

One advantage of running the algorithm for the metric in the cameras is that it becomes possible to access a good enough reference image and, thus, use full reference methods.

The metrics can be further split up into two broad categories: phenomenological, that analyze features of the image without attempting to understand the fundamentals of the interaction between the human visual system and the image, and model-based, that incorporate some more or less accurate representation of the human visual system in the assessment algorithm [Zhai and Min, 2020].

However, the aim of the research in this thesis is not to develop or use the best image quality metric available but rather to use a very simple metric, i.e., the qp value, in order to assess streaming videos quality in a short time.

4

Control Background

In this chapter we introduce the control principles used in Papers II, III and IV. We start by explaining the basics of PID control and tracking-based anti-windup before moving to ways of combining controllers with cascade control and mid-ranging.

4.1 PID control

Proportional-Integral-Derivative (PID) control is one of the most common control algorithms used in industry. The PID algorithm consists of three basic terms: the proportional (P) term, the integral (I) term and the derivative (D) term [Åström and Murray, 2008]. PID controllers use the control error (denoted e in Fig. 4.1) as the input in order to calculate the value of the control signal (denoted u in Fig. 4.1), i.e., the input to the system or process that is under control. The control error is the difference between the setpoint, i.e., the desired value of the output of the process, and the actual measured value of the output. The reason for the control could be to mitigate disturbances acting on the process or to modify the dynamics of the process, e.g., to speed up the response to a change in the setpoint. The time-domain equation for

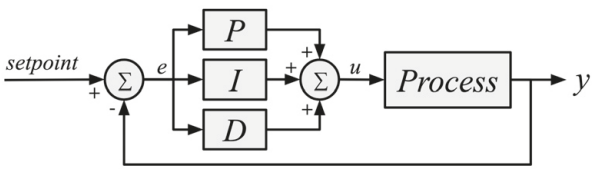


Figure 4.1 Simplified PID architecture.

the text-book version of the PID controller is given by Equation 4.1

$$\begin{aligned} u = &\quad \mathrm{P} \quad + \quad \mathrm{I} \quad + \quad \mathrm{D} \\ u(t) = &K\left(e(t) + \frac{1}{T_i}\int^t e(s)\,ds + T_d\frac{de}{dt}\right). \end{aligned} \tag{4.1}$$

The proportional term depends only on the control error and multiplies this error by the proportional gain value, K, of the controller, i.e., the P-term depends on the current value of the error. The integral term sums the error over time. This term depends on the past value of the error. Even a small error will cause the integral term to increase slowly. The integral term will continue to change over time unless the error is zero, hence, the effect of the integral term is to drive the steady state error to zero. The derivative term instead tries to predict what the error will be at horizon into the future. This is done using a simple linear extrapolation.

It is sometimes to instead use the frequency-domain version of the PID controller given by Equation 4.2

$$U(s) = K\left(E(s) + \frac{1}{T_i s}E(s) + T_d E(s)\right), \tag{4.2}$$

where s is the Laplace operator. Using this notation $1/s$ corresponds to the integrator.

PI control, i.e., PID without the derivative term, is often suitable for processes of low order and/or when only a very coarse model of the process is available and the control specifications are not so challenging. This is often the case with applications of control to computer and communication systems. This is the reason why PI control is the method of choice in Paper II and III, and as a part of the price discrimination scheme in Paper IV.

The code for the continuous-time PI controller when the I-term is discretized using a 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;
u = max(u_low, min(v, u_max));
writeU(u);
I = I + (K/Ti)*e;
```

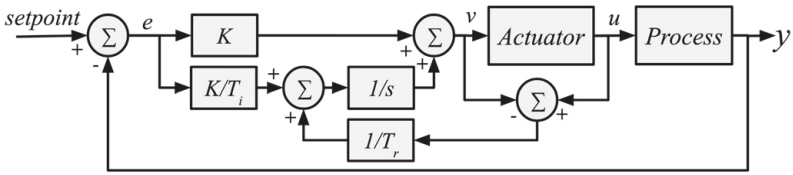


Figure 4.2 A PI controller with tracking-based anti-windup.

4.2 Tracking-based anti-windup

All actuators have physical limitations or saturations, e.g., a motor has limited maximum velocity or a control valve cannot be more than fully open. The classic PID controller is not aware of these actuator limitations and could easily generate a control signal that is outside the bounds.

When the actuator saturates the feedback loop will be broken because the output of the actuator is then not influenced by its input. Any unstable modes in the controller (modes where the control system loses the ability to provide a bounded output when a bounded input is applied to it) may then drift, or windup, to very large values. The most common case of this is the integral term. When the control signal is limited the integrator will grow to a potentially very large value, creating a large overshoot (where the process exceeds the target value). Since the value of the integrator is so large, it may take considerable time before it has decreased again when the error becomes negative, causing a subsequent large undershoot.

Integrator windup can be avoided by making sure that the integral is kept to a proper value when the actuator saturates. A common anti-windup scheme that achieves this is known as tracking or back-calculation. It consists of a feedback loop inside the controller that adjusts the value of the integral term so that the output of the controller tracks the saturated value, see Fig. 4.2. The difference between actuator input and output is fed back to the integrator. As soon as the actuator saturates, this signal becomes non-zero and prevents the integrator from winding up. One could add a tracking time constant T_r to this feedback to influence how fast the integrator should be adjusted.

The method for handling global resource constraints proposed in Paper II is inspired by tracking-based anti-windup. The idea behind the method is to modify the integrators of the individual PI controllers so that the sum of the control signals does not exceed a maximum value.

4.3 Cascade control

Cascade control is used when there are more than one measurement available to a controller, but only one control variable to actuate on. Cascade control

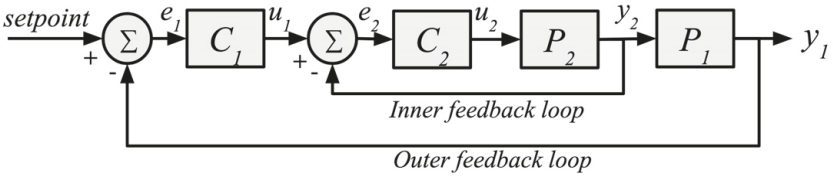


Figure 4.3 Cascade control architecture (C_1 and C_2 being outer and inner controllers of processes P_1 and P_2).

involves the use of two controllers, with the output of the first, or outer, controller providing the setpoint for the second, or inner controller. In the resulting structure the inner feedback loop is nested inside the outer feedback loop. This is illustrated in Figure. 4.3. Normally the inner loop is designed to be faster than the outer loop. The advantage of cascade control is that it can respond more quickly to disturbances acting on the inner loop part of the process than would be the case if only the outer controller was used. The design of the outer controller also becomes simpler since the inner controller will simplify the dynamics of the process.

Cascade control is used in the control architecture for controlling storage and network resources presented in Paper III.

4.4 Mid-ranging

Mid-range control can be considered the dual of cascade control, where two control signals are used to control one measurement signal. A practical example could be to control the flow of a liquid using two control valves according to Fig. 4.4 [Åström and Hägglund, 2006; Hägglund, 2021]. 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)* according to Fig. 4.5. The VPC uses the control signal of v_1 (u_1 in Fig.4.5) as the measurement signal, and selects the reference signal of v_2 (r_2 in Fig.4.5)

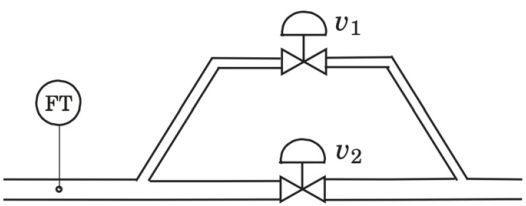


Figure 4.4 Mid-ranging application example.

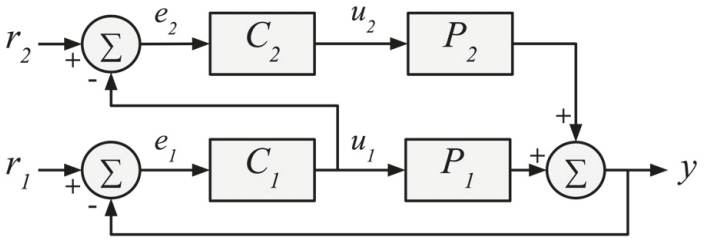


Figure 4.5 Mid-ranging controller structure.

to ensure that the control signal of v_1 lies in the middle of its operating range, i.e., at 50%. Process P_1 and controller C_1 together form a fast feedback loop. The valve position controller C_2 controls the valve position of controller C_1 via the controller output u_2.

The advantage of mid-ranging is to avoid relying on a single expensive valve with both large control authority and high resolution and instead replace it with two cheaper valves with different characteristics which together achieve the same result.

Mid-ranging is the basis for the control architecture for controlling storage and network resources presented in Paper III, where it is used to combine the storage and network bandwidth reservation controllers. This structure presents the advantage of allowing finer control of the process with two simple controllers reacting at different speeds where C_1 has a short response time and C_2 has a slower response time.

4.5 Bumpless mode changes

Controllers are dynamic systems. This means that when switching between controllers or between different operating modes of the same controller, e.g., between manual and automatic mode, we need to ensure that the internal state of the controller that is switched in corresponds to the internal state of the controller that is switched out. If that is not the case the switching will cause bumps in the control signal. There are multiple ways to ensure this depending on the controller and implementation used. For PID controllers on the form presented in Section 4.1, tracking could be used to ensure that the integrator, i.e., the internal state, of the controller has the correct value at the time of the switching.

Tracking-based bumpless mode changes could (and should if time had permitted) have been used in Paper III to ensure bumpless switches between the two main operating modes of the cameras.

5

Game Theory Background

In this chapter we introduce the pricing and auction theory principles used in Papers IV and V. We start with price discrimination, an important economics theory used in Paper IV. We then explain the Vickrey-Clarke-Groves auction mechanism and its associated optimization problem, the knapsack problem, which are used in Paper V.

5.1 Price discrimination

Price discrimination is a selling strategy used in economics. The principle is to charge buyers different prices for the same product or service based on what the sellers think they can get the buyers to accept [Krugman and Obstfeld, 2018]. In a perfect market, the seller charges each buyer the maximum price they are willing to pay. In more common forms, the seller groups buyers based on certain attributes and charges each group a different price. Price discrimination is practiced based on the seller's belief that buyers in certain groups can be asked to pay more or less based on certain characteristics or on how they value the product at a certain time.

We can identify three degrees of price discrimination:

- First-degree discrimination (or perfect price discrimination): The seller charges the maximum possible price for each unit sold to buyers. Because prices vary among units, the seller captures all available buyer surplus for itself [Frank and Cartwright, 2010].
- Second-degree price discrimination: The seller charges a different price for different quantities sold, such as quantity discounts on bulk purchases [Frank and Cartwright, 2010].
- Third-degree price discrimination: Here the seller charges a different price to different buyer groups. This discrimination is the most common [Bergemann et al., 2015].

Third-degree price discrimination is the basis for the resource management method presented in Paper IV.

5.2 Vickrey-Clarke-Groves auctions

A Vickrey-Clarke-Groves auction mechanism is the method used in Paper V to split the storage resource among cameras as well as determine the payment requested for these resources. A Vickrey-Clarke-Groves (VCG) auction is a type of sealed-bid auction of multiple items (see Section 2.2). Bidders submit bids that reflect their valuations for the items to the seller/auctioneer, without knowing the bids of the other participants. The auction system assigns the items in a socially optimal manner, i.e., it charges each individual based on the harm (or loss) they cause to other bidders.

The VCG auction mechanism gives bidders an incentive to bid their true valuations of the items, by ensuring that the dominant strategy for each bidder is to reveal their true valuations. This mechanism can be undermined by bidder collusion (when buyers group collaborate).

The VCG auctions are named after William Vickrey [Vickrey, 1961], Edward H. Clarke [Clarke, 1971] and Theodore Groves [Groves, 1973] for their papers which successively generalized the VCG auction mechanism.

Consider an auction where a set of identical indivisible products are being sold (the auctioneer having N items for sale). At auction start, each bidder announces to the auctioneer (and only to him) the maximum price they are willing to pay to receive n of the N items (with $0 < n \leq N$), i.e., the bid is a combination of number of items and an associated price. Each bidder is allowed to declare more than one bid, since its willingness-to-pay per unit might be different depending on the total number of items it receives. When a predefined time has passed or all the bids are received, the auction is closed.

All the possible combinations of bids are then considered by the auction system and the one maximizing the total amount of money is selected, on the conditions that the number of items allocated does not exceed N and that at most one bid per bidder is selected. Bidders who have been assigned items will receive them but the price they pay in exchange is not the amount they had bid initially but the marginal harm their bid has caused to other bidders (which is at most as high as their original bid).

If the sum of bids of the second best combination of bids is the same as that of the best combination, then the price paid by the buyers will be the same as their initial bid, otherwise the price paid will be lower. In that sense the VCG auctions can be viewed as second-price auctions.

The marginal harm caused to other participants can be calculated as the sum of bids of the auction from the best combination of bids (the decided allocation) excluding the participant under consideration. This can be done

by solving a knapsack problem per bidder where a selected bidder is not present and see what would the allocation be in that case. The difference between the selected allocation prices with all bidders and the one where a considered bidder wouldn't be present provides us the marginal harm value of that bidder.

If bidders are rational (and there is no collusion), we can assume that the provided price is equal to the real willingness to pay since only the marginal harm to other bidders will be charged to each participant, making truthful reporting a weakly-dominant strategy. This type of auction incites truthfulness and maximizes the total welfare but will not maximize the seller's revenue.

Practical illustration

We can take a practical example with a pastry chef selling identical "semlor", a famous Swedish pastry, which shall not be split between people as it is considered an utter lack of manners. The bidders would pay the semla/semlor in Swedish kronor (kr).

If for example we consider 3 bidders and $N = 2$ semlor for sale:

- Bidder A wants only 1 semla and is willing to pay 50kr for it.
- Bidder B wants 1 semla and would pay up to 40kr for it.
- Bidder C wants only 2 semlor and is willing to pay 70kr to get them.

The outcome of the auction is determined by maximizing the amount of money from the bids which is a done by solving a knapsack problem. The two semlor go to bidder A and bidder B, since their combined bid of $50\text{kr} + 40\text{kr} = 90\text{kr}$ is greater than the bid for two semlor made by bidder C who is willing to pay only 70kr. Other knapsack problems are solved to calculate the price each bidder will have to pay for the item.

As for the payments:

- If bidder A was not present, the two semlor would both be allocated to bidder C (as $70\text{kr} > 40\text{kr}$). The total loss of welfare (or harm) inflicted by not having bidder A present is then $90\text{kr} - 70\text{kr} = 20\text{kr}$. The price A would pay for its semla is $50\text{kr} - 20\text{kr} = 30\text{kr}$.
- The same logic applies for bidder B who would need to pay $40\text{kr} - (90\text{kr} - 70\text{kr}) = 20\text{kr}$.
- Finally, the payment for bidder C is $(50\text{kr} + 40\text{kr}) - (50\text{kr} + 40\text{kr}) = 0\text{kr}$. (since bidder C did not receive anything, he should not pay anything).

At the end of the auction, the total utility has been maximized since all the semlor have been attributed to the bidders with the highest combined willingness-to-pay.

Knapsack optimization

The knapsack problem is a problem in combinatorial optimization [Salkin and De Kluyver, 1975]. The problem is defined as follows. Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit, and the total value is as large as possible. One can visualize it by trying to fill a knapsack with a limited space with the most valuable combination of items.

The most common problem being solved is the 0-1 knapsack problem where items are indivisible and unique (meaning they can only be taken once). The knapsack problem is, despite its apparent simplicity, a NP-hard problem and thus no fast scalable and exact methods exists to solve it. Several approaches can be used to solve knapsack problems, e.g, dynamic programming or branch and bound. In Paper V, the Google OR-Tool library which utilizes dynamic programming is used.

The optimization problem used to solve a VCG auction allocation can be written as:

$$\begin{aligned} &\text{maximize } \textstyle\sum_{i=1}^{n} v_i \cdot x_i \\ &\text{subject to } \textstyle\sum_{i=1}^{n} w_i \cdot x_i \leq W \quad \text{and} \quad x_i \in \{0,1\} \end{aligned} \tag{5.1}$$

with n being the number of items indexed by i ($1 \leq i \leq n$), v_i being the value of item i, w_i its associated weight, W the maximum total weight, and x_i being a variable that indicates if the item was selected or not.

Practical illustration

Imagine that Olivier, a Frenchman, goes out for a picnic. He only has a small fabric knapsack (or backpack) which can hold at most 3kg before it breaks. Olivier would like to get a lot of food but he needs to choose. He goes shopping and finds a collection of items i which he would enjoy (value v) but has a weight (w):

- Bottle of red wine, $v = 5$ and $w = 1$kg.
- Bottle of water, $v = 1$ and $w = 1$kg.
- Camembert cheese, $v = 5$ and $w = 1$kg.
- Comté cheese, $v = 2$ and $w = 2$kg.
- Chicken sandwich $v = 2$ and $w = 1$kg.
- Flan cake, $v = 5$ and $w = 1$kg.
- Brownie cake, $v = 1$ and $w = 1$kg.

Which items should Olivier take for his picnic? The wine, the Camembert cheese and the flan cake of course! Because the sum of their weights is 3kg but the total value is 15. Any other combination within 3kg would provide a lower value to Olivier.

6

Conclusions and Future Work

In this thesis we propose different methods to efficiently allocate storage and bandwidth resources for video surveillance camera systems. These solutions, each trying to solve a similar problem, use different techniques from automatic control, economics theory and game theory.

A proper estimation of the video frame sizes is the key to properly dimension the network infrastructure for real-time video-surveillance systems. For this, we presented in Paper I how to derive upper bound estimates for the size of the frames in a video streaming system. We discussed which characteristics influence the bandwidth requirements of different cameras and derived models for the upper bound estimates of the size of frames. This allowed us to precisely formulate the problem of allocating network bandwidth to a set of cameras in a switched Ethernet network environment.

We then used the model described in Paper I to simulate video frame sizes used in Paper II. In Paper II we proposed a method for enforcing soft resource constraints in camera networks where PI controllers were used to decide the amount of disk space resources that each camera should obtain for storing the video stream. These resource constraints were expressed as a global limitation on the sum of the control signals. The method was inspired by tracking-based anti-windup for PID control. The approach was evaluated in simulation with very good results.

The system from Paper II was then further extended in Paper III where a control system architecture was proposed that combined mid-ranging and global resource tracking. The architecture was used to control, assign, and prioritize storage and bandwidth reservation resources in camera networks.

In Paper IV we then explored pricing theory by proposing a lightweight method based on price discrimination of storage costs for system-level optimization of video quality. This allowed camera systems to have an equivalent

video quality over the whole system with little overhead and very little information about each device.

We finally used auction theory methods in Paper V by proposing a method for the allocation of storage space resources based on the VCG mechanism and a utility derived from the compression level and its variation. The main advantage of the latter is that the storage provider is not required to be aware of the cameras parameters and still is able to efficiently allocate the storage based solely on the cameras' declared values.

The control, pricing and auction theory solutions proposed illustrate different possible ways to solve the storage and bandwidth allocation problem. The control solutions presented give very good results in the case of joint bandwidth/storage allocation but require each device in the system to be cooperating for the solution to work as expected. The price discrimination approach is very interesting as it is lightweight and can be implemented easily in a non-cooperating environment, provided that there is no encryption of the video streams which is less and less likely nowadays. The VCG auction mechanism approach is probably the most future proof approach of all. Even if is more computationally heavy (which would limit the number of devices handled in the system), it provides guarantees in the case of competitive systems. It can work with encrypted content as we do not require to look at the streamed content and allows for devices with different optimization goals to share the same pool of resources. The VCG mechanism should provide a solid basis for more complex and smarter valuation of multiple resources while allowing the different devices to seek different amounts of resources for different use cases without having to redesign the whole mechanism which would be needed in the other proposed solutions.

6.1 Future Work

There are several directions in which the results presented in the thesis could be extended and further evaluated. Here this is outlined for each paper.

The model developed in Paper I should be evaluated against more test scenarios in order to refine the upper-bound estimation in other network contexts and use it for bandwidth scheduling applications.

The formal properties of the approach in Paper II need further study to, e.g., analyze the multiple equilibria that may occur. The approach can also be modified in different ways, e.g., so that controllers without any integral term can be used. Also the delay between the storage set-point request and allocation should be introduced and studied.

In Paper III, performance can be further improved by introducing decoupling in the mid-range controllers. Additional changes should be introduced to handle saturation, the saturation feedback could be directed to the band-

width controllers to improve the saturation case. We should also implement bumpless mode changes when changing between the two main controller modes. The network model could also be made more realistic. 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 bandwidths.

A logical extension of Papers IV and V would be to handle multiple storage providers and develop more complex utility functions for both cameras and storage sellers which would take into account different constraints such as network latency and bandwidth. The work should also be expanded with multiple inter-dependent resources, e.g., bandwidth, storage, and CPU, etc. We should also study to handle prioritization based on priority or task allocation of cameras. Implementing such systems in real large-scale systems would allow to study their scalability and behaviours and complete the simulations provided.

For Paper IV we could use a different convergence method which would optimize the storage usage more by allowing the compression levels to slightly deviate for some cameras.

A pertinent future approach would also be to use an application-specific metric or a recognized quality metric such as the structural similarity index measure (SSIM), peak signal-to-noise ratio (PSNR) or other metrics as described in [Yang, 2007].

These solutions should also be implemented in real physical camera systems using different network architectures and cameras with different characteristics to see how it will affect the resource allocation mechanisms.

Finally, more complex camera and environment knowledge through learning could be incorporated. Adding extra knowledge in the valuation of resources and the decision process of cameras would allow to take advantage of the dynamic nature of distributed systems. This would enable self-aware and self-expressive systems [Bellman et al., 2020; Esterle et al., 2017; Becker et al., 2012; Rinner et al., 2012].

Bibliography

Ahad, A., M. Tahir, M. Aman Sheikh, K. I. Ahmed, A. Mughees, and A. Numani (2020). "Technologies trend towards 5G network for smart healthcare using iot: a review". *Sensors* **20**:14, p. 4047.

Åström, K. J. and T. Hägglund (2006). *Advanced PID control*. ISA, Advanced PID control.

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

Becker, T., A. Agne, P. R. Lewis, R. Bahsoon, F. Faniyi, L. Esterle, A. Keller, A. Chandra, A. R. Jensenius, and S. C. Stilkerich (2012). "Epics: engineering proprioception in computing systems". In: *2012 IEEE 15th International Conference on Computational Science and Engineering, Paphos, Cyprus*, pp. 353–360.

Bellman, K., C. Landauer, N. Dutt, L. Esterle, A. Herkersdorf, A. Jantsch, N. TaheriNejad, P. R. Lewis, M. Platzner, and K. Tammemäe (2020). "Self-aware cyber-physical systems". *ACM Transactions on Cyber-Physical Systems* **4**:4.

Bergemann, D., B. Brooks, and S. Morris (2015). "The limits of price discrimination". *American Economic Review* **105**:3, pp. 921–57.

Brunnström, K., A. Djupsjöbacka, and B. Andren (2021). *Objective video quality assessment methods for Video assistant refereeing (VAR) Systems*. Tech. rep. RISE report 2021:30. RISE Research Institute of Sweden AB.

Camacho, E. F. and C. B. Alba (2013). *Model Predictive Control*. Springer science & business media.

Charilas, D. E. and A. D. Panagopoulos (2010). "A survey on game theory applications in wireless networks". *Computer Networks* **54**:18, pp. 3421–3430.

Chen, Y., K. Wu, and Q. Zhang (2014). "From QoS to QoE: a tutorial on video quality assessment". *IEEE Communications Surveys & Tutorials* **17**:2, pp. 1126–1165.

Clarke, E. H. (1971). "Multipart pricing of public goods". *Public choice* **11**, pp. 17–33.

Dost, S., F. Saud, M. Shabbir, M. G. Khan, M. Shahid, and B. Lovstrom (2022). "Reduced reference image and video quality assessments: review of methods". *EURASIP Journal on Image and Video Processing* **2022**:1, pp. 1–31.

Edpalm, V., A. Martins, M. Maggio, and K.-E. Årzén (2018). *H.264 Video Frame Size Estimation*. Tech. rep. Technical Reports TFRT-7654. Department of Automatic Control, Lund Institute of Technology, Lund University, Lund, Sweden.

Esterle, L., P. R. Lewis, R. McBride, and X. Yao (2017). "The future of camera networks: staying smart in a chaotic world". In: *Proceedings of the 11th International Conference on Distributed Smart Cameras (ICDSC)*. Association for Computing Machinery, Stanford, CA, USA, pp. 163–168.

Frank, R. H. and E. Cartwright (2010). *Microeconomics and behavior*. Vol. 8. McGraw-Hill New York.

Groves, T. (1973). "Incentives in teams". *Econometrica: Journal of the Econometric Society*, pp. 617–631.

Hägglund, T. (2021). "A feedforward approach to mid-ranging control". *Control Engineering Practice* **108**:March, pp. 1–10.

IPVM (2021). *Top 5 problems in video surveillance storage*. URL: https://ipvm.com/reports/problems-video-surveillance-storage (visited on 2021-08-18).

ISO/IEC MPEG & ITU-T VCEG, J. V. T. (of (2003). *Draft itu-t recommendation and final draft international standard of joint video specification (itu-t rec. h.264/iso/iec 14496-10 avc)*. URL: https://www.loc.gov/preservation/digital/formats/fdd/fdd000081.shtml (visited on 2022-01-10).

ITU-T (2004). *Objective perceptual video quality measurement techniques for digital cable television in the presence of full reference (itu-t rec. j.144)*. URL: https://www.itu.int/rec/T-REC-J.144-200403-I (visited on 2022-03-01).

ITU-T (2010). *H.264 standard documentation*. URL: https://www.itu.int/rec/T-REC-H.264 (visited on 2019-06-13).

Kitchin, R. (2015). "The promise and peril of smart cities". *Computers and law: the journal of the Society for Computers and Law* **26**:2.

Krishna, V. (2009). *Auction Theory*. Academic press.

Krugman, P. R. and M. Obstfeld (2018). *International trade: Theory and policy*. Pearson.

Lewis, P. R., L. Esterle, A. Chandra, B. Rinner, and X. Yao (2013). "Learning to be different: heterogeneity and efficiency in distributed smart camera networks". In: *2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), Philadelphia, USA*, pp. 209–218.

Li, J., H. Chen, Y. Chen, Z. Lin, B. Vucetic, and L. Hanzo (2016a). "Pricing and resource allocation via game theory for a small-cell video caching system". *IEEE Journal on Selected Areas in Communications* **34**:8, pp. 2115–2129.

Li, Z., A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara (2016b). *Toward a practical perceptual video quality metric*. URL: https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652 (visited on 2022-03-01).

Lin, X., H. Ma, L. Luo, and Y. Chen (2012). "No-reference video quality assessment in the compressed domain". *IEEE Transactions on Consumer Electronics* **58**:2, pp. 505–512.

Lindberg, M. (2007). *A Survey of Reservation-Based Scheduling*. Tech. rep. Department of Automatic Control, Lund Institute of Technology, Lund University. URL: https://lup.lub.lu.se/search/files/4381632/8627270.pdf.

Milgrom, P. (2004). *Putting auction theory to work*. Cambridge University Press.

Nisan, N. and A. Ronen (2007). "Computationally feasible VCG mechanisms". *Journal of Artificial Intelligence Research* **29**, pp. 19–47.

Niyato, D., N. C. Luong, P. Wang, and Z. Han (2020). "Auction theory for computer networks".

Osborne, M. J. and A. Rubinstein (1994). *A course in game theory*. MIT press.

Owen, G. (2013). *Game theory*. Emerald Group Publishing.

Pang, Y.-C., S.-L. Chao, G.-Y. Lin, and H.-Y. Wei (2014). "Network access for m2m/h2h hybrid systems: a game theoretic approach". *IEEE Communications Letters* **18**:5, pp. 845–848.

Rajkumar, R., L. Sha, and J. P. Lehoczky (1988). "Real-time synchronization protocols for multiprocessors." In: *IEEE Real-Time Systems Symposium (RTSS), Huntsville, USA*. Vol. 88, pp. 259–269.

Rinner, B., B. Dieber, L. Esterle, P. R. Lewis, and X. Yao (2012). "Resource-aware configuration in smart camera networks". In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 58–65.

Salkin, H. M. and C. A. De Kluyver (1975). "The knapsack problem: a survey". *Naval Research Logistics Quarterly* **22**:1, pp. 127–144.

Sha, L., R. Rajkumar, and J. P. Lehoczky (1990). "Priority inheritance protocols: an approach to real-time synchronization". *IEEE Transactions on computers* **39**:9, pp. 1175–1185.

Shahid, M., A. Rossholm, B. Lövström, and H.-J. Zepernick (2014). "No-reference image and video quality assessment: a classification and review of recent approaches". *EURASIP Journal on Image and Video Processing* **2014**:1, pp. 1–32.

Shin, K. and Y.-C. Chang (1995). “A reservation-based algorithm for scheduling both periodic and aperiodic real-time tasks”. *IEEE Transactions on Computers* **44**:12, pp. 1405–1419.

Taylor, G. A., K. K. Tsui, and L. Zhu (2003). “Lottery or waiting-line auction?” *Journal of Public Economics* **87**:5-6, pp. 1313–1334.

Teo, P. and D. Heeger (1994). “Perceptual image distortion”. In: IEEE International Conference on Image Processing, Austin, TX, 982–986 vol.2.

Vickrey, W. (1961). “Counterspeculation, auctions, and competitive sealed tenders”. *The Journal of Finance* **16**:1, pp. 8–37.

Wang, Z., A. Bovik, H. Sheikh, and E. Simoncelli (2004). “Image quality assessment: from error visibility to structural similarity”. *IEEE Transactions on Image Processing* **13**:4, pp. 600–612.

Wei, W., X. Fan, H. Song, X. Fan, and J. Yang (2016). “Imperfect information dynamic stackelberg game based resource allocation using hidden markov for cloud computing”. *IEEE Transactions on Services Computing* **11**:1, pp. 78–89.

Wiegand, T., G. J. Sullivan, G. Bjontegaard, and A. Luthra (2003). “Overview of the h.264/avc video coding standard”. *IEEE Transactions on Circuits and Systems for Video Technology* **13**:7, pp. 560–576.

Wittenmark, B., K. Åström, and K.-E. Årzén (2003). *Computer Control: An Overview*. Tech. rep. Department of Automatic Control, Lund University.

Xue, Y., Y.-F. Ou, Z. Ma, and Y. Wang (2010). “Perceptual video quality assessment on a mobile platform considering both spatial resolution and quantization artifacts”. In: *2010 18th International Packet Video Workshop*. IEEE, pp. 201–208.

Xue, Y., Y. Song, Y.-F. Ou, and Y. Wang (2013). “Video adaptation considering the impact of temporal variation on quantization stepsize and frame rate on perceptual quality”. In: *International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, pp. 70–74.

Yang, K.-C. (2007). *Perceptual quality assessment for compressed video*. PhD thesis. UC San Diego.

Zhai, G. and X. Min (2020). “Perceptual image quality assessment: a survey”. *Science China Information Sciences* **63**:11, pp. 1–52.

Paper I

Camera Networks Dimensioning and Scheduling with Quasi Worst-Case Transmission Time

Viktor Edpalm Alexandre Martins, Karl-Erik Årzén

Martina Maggio

Abstract

This paper describes a method to compute frame size estimates to be used in quasi Worst-Case Transmission Times (qWCTT) for cameras that transmit frames over IP-based communication networks. The precise determination of qWCTT allows us to model the network access scheduling problem as a multiframe problem and to re-use theoretical results for network scheduling. The paper presents a set of experiments, conducted in an industrial testbed, that validate the qWCTT estimation. We believe that a more precise estimation will lead to savings for network infrastructure and to better network utilization.

1. Introduction

In the modern interconnected world, multiple devices share access to networking resources, such as transmission bandwidth. For some of these devices — e.g., video surveillance cameras connected to a monitoring station [Rinner and Wolf, 2008] — access to networking resources is often more critical than access to computing resources [Vandalore et al., 2001; Wang et al., 2008; Veeraraghavan and Weber, 2008]. Scheduling network access is therefore crucial for the satisfaction of real-time requirements [Toka et al., 2017; Almeida et al., 2007; Pedreiras and Almeida, 2003; Tang et al., 2017], like the timely transmission of surveillance videos from different cameras [Seetanadi et al., 2017].

A typical video surveillance system comprises of multiple cameras disseminated over an area. These cameras continuously record a specific scene, it being an office space, a parking lot, a road, or any other alternative. The recorded scenes are of course different from one another, but their characteristics do not evolve significantly over time. A camera that is installed outdoor in a parking lot will record similar scenes, mostly involving cars and people, in different light conditions. At the same time, a camera that is pointing to a highway lane will (most likely) record either an empty road, or the passage of cars. A common challenge in the video surveillance industry is to tailor the entire infrastructure of the surveillance system to achieve a certain level of quality, while keeping the cost as limited as possible. Today, the video industry is mainly focused on using IP cameras, which stream videos that are compressed using the H.264 standard. In order to tailor the infrastructure, one must be able to anticipate how much data each camera in the system is expected to produce, given its unique set of internal characteristics and settings — e.g., position, placement, surrounding environment, etc. Such an estimate can be conservative, assuming that video frames are not compressed. Currently, conservative techniques are adopted for practical applications [Wang et al., 2008; Almeida et al., 2007; Pedreiras and Almeida, 2003; Seetanadi et al., 2017]. However, conservativeness greatly increases infrastructure cost and limits the network usage. Non-conservative estimates have the potential of reducing the operational cost of video-surveillance systems. The challenge explored in this paper is therefore the estimation of the amount of data produced by each camera in the surveillance system.

We motivate our investigation by drawing a parallel between network scheduling for video surveillance camera systems and CPU scheduling. Using the periodic task model, a set $\mathcal{T} = \{\tau_1, \ldots, \tau_p\}$ of p tasks execute on a given hardware platform. Each $\tau_i = \{E_i, P_i, D_i\}$ is activated at precise time instants, determined by the period P_i, and must meet a given deadline D_i. For scheduling policies to be effective at ensuring the satisfaction of deadline constraints in complex systems, schedulers use information about the Worst-Case Execution Time (WCET) E_i of a task τ_i on the given hardware.

Similarly, a set $\mathcal{C} = \{c_1, \ldots, c_p\}$ of p surveillance cameras transmits video streams to a monitoring station. Each camera c_i has a given frame rate f_i, denoting the number of frames that the camera captures in a second. The frame rate has a direct implication on the transmission requirements of the camera, its inverse $1/f_i$ being equal to the activation period. For simplicity, we can assume that the deadline to transmit the currently captured frame is equal to the period. Hence, in this setting, reusing well-known CPU scheduling algorithms for network access depends on de-

termining the Worst-Case Transmission Time (WCTT) for video frames. From the theoretical perspective, the task set model is not as simple as a set of periodic tasks, and can be described using a multiframe model [Mok and Chen, 1997], as will be shown in the following. Also, video encoders are very complex and the frame size depends heavily on the encoded scene. We therefore cannot compute precise upper bounds — e.g., using static analysis or formal methods — that guarantee that the given size is never exceeded. We therefore limit ourselves to the computation of quasi Worst-Case Transmission Times (qWCTT). We have experimentally verified that our estimate of the upper bound is valid in most cases and we have not encountered any case in which a frame exceeding our estimated upper bound is not a result of software bugs.

This paper contributes to the state of the art of real-time systems (and real-time surveillance video streaming) by:

- Determining a combination of measurable parameters that can accurately predict the expected H.264 frame sizes;
- Computing reasonable estimates of upper bounds for the qWCTT of frames of different types over a network, casting the problem of scheduling switched Ethernet network access into a multiframe non-preemptive scheduling problem;
- Conducting a thorough experimental campaign to validate our findings and the given models, providing parameters for different camera models and circumstances and allowing researchers to use the derived models to verify real-time properties on the network transmission time.

From the industrial perspective, the relevance of this paper is in enabling infrastructure tailoring for a video surveillance system and selecting quantities like the total required network bandwidth to guarantee a given video stream quality.

In the following, we review the H.264 standard and terminology in Section 2. Section 3 then discusses our models; enumerating the parameters, explaining how to measure them when needed, and showing the equations used to determine the frame sizes. Section 4 presents related efforts and Section 5 shows experimental results obtained with 6 different cameras in a laboratory environment and 24 different real-life surveillance scenarios. We finally conclude the paper in Section 6.

2. Background on Video Encoding

This section provides a brief overview of H.264, also called MPEG-4 part 10 AVC, which currently is the *de facto* standard for video encoding and decoding[1]. Table 1 presents a recap of the acronyms used in the paper.

H.264 is a video compression standard that defines how a video should be decoded. The implementation of the encoding is left to the manufacturer's discretion. The

[1] The first official version H.264 version was approved in March 2003 [ISO/IEC MPEG & ITU-T VCEG, 2003; Wiegand et al., 2003] and has since evolved over time. The standard now includes more features and modes, the latest version being approved in April 2017 [ITU-T, 2017]. The MPEG LA organization administers most of the licenses for patents applying to this standard.

Table 1. Nomenclature: Acronyms.

Acronym	Brief Explanation
GOP	**Group of Pictures:** Set of one I-frame and multiple P- and B-frames. The number also represents the amount of frames between two consecutive I-frames
HDR	**High Dynamic Range:** Technique used to enhance video, that typically allows frames to include more details and be sharper
IDR	**Instantaneous Decoding Refresh:** I-frame that imposes a refresh, i.e., following frames must not need any information from frames prior to the IDR I-frame
QP	**Quantization Parameter:** Compression parameter defined in the H.264 standard, higher numbers indicate more information loss
SAO	**Size of Average Object:** Reflects the expected distance to an object in an image, determined by factors like the zoom level, field of view, and lens type, as well as placement of the camera
WCET	**Worst-Case Execution Time:** Upper bound on the time it takes for a task to execute on a given hardware platform
WCTT	**Worst-Case Transmission Time:** Indicates the maximum time it takes to transmit a frame of the video using the available network bandwidth
qWCTT	**quasi Worst-Case Transmission Time:** Indicates a non-exact upper bound for the transmission time of a frame using the available network bandwidth

standard describes a *block based hybric codec*, i.e., a video is decomposed in blocks of data for encoding. To allow for video compression, H.264 uses motion-compensated encoding, i.e., it describes a frame by referencing parts of other frames, thus capturing the motion of objects across different frames [Wiegand et al., 2003]. A stream encoded with H.264 contains a sequence of frames, these frames are not necessarily encoded following the display order or time they were captured. Based on the frame encoding, it is possible to distinguish between three different types of frames: Intra frames (I-frames), Predicted frames (P-frames), and Bi-directional predicted frames (B-frames).

- I-frames are (usually[2]) self-contained. An I-frame contains the full image and does not need additional information in the decoding process. In terms of encoding, these are fast and easier to encode, as all the information should be present in the resulting frame and no extra buffer containing other frames are necessary. In terms of size, on the contrary, these are the most space-consuming type of frames.
- P-frames are encoded using information contained in the current frame and in previous ones (up to the last self-contained I-frame). In the encoding of a P-frame, part of the image can be encoded using references to previous ones with extra information to reproduce the difference, instead of repeating the information. This allows the encoder to compress the frame, reducing its size, at the cost of additional computation and buffering.

[2] If an I-frame is marked as an Instantaneous Decoding Refresh (IDR), its encoding is self-contained. In most cases, this is true, but there are certain conditions in which this does not hold. Since we are interested in estimating the upper bounds, we can safely assume that the upper bound of an I-frame is self-contained.

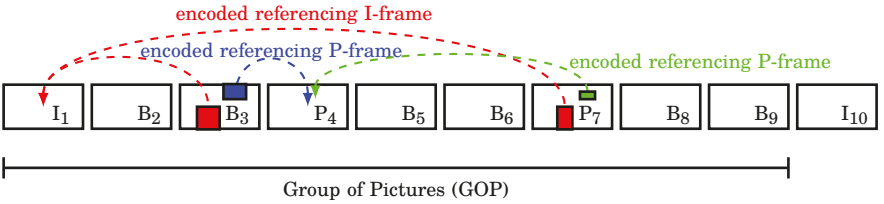


Figure 1. H.264 frame sequence: I-frames, P-frames, B-frames, and Group of Pictures.

- B-frames are encoded using both information from previous frames and information from following frames. In a B-frame, the encoder can introduce references to frames that come next, in display order, with respect to the current one being encoded. B-frames require the most computational capacity for the encoding, but are usually the lightest in terms of space consumption.

Figure 1 shows a sequence of 10 frames. The first nine frames in the example denote a Group of Pictures (GOP). A GOP consists of an I-frame followed by a sequence of B-frames and P-frames. The I-frame can be marked as an Instantaneous Decoding Refresh (IDR), meaning that the following frames do not need information from frames prior to that one in the sequence. If all the I-frames are marked as IDR points, the decoding of each GOP is independent, otherwise it is not. The sequence of frame types is determined and fixed by a high-level controller before the frame encoding starts.

For the sequence shown in Figure 1, the first and the last frame are encoded as I-frames. The fourth and the seventh are encoded as P-frames. The remaining ones are encoded as B-frames. The red arrows in the Figure indicate areas of the third and seventh frames – respectively a B-frame and a P-frame – that are encoded as references to the previous I-frame. The blue arrow shows an area of the third B-frame that is encoded as reference to the following P-frame. The green arrow shows an area of the seventh P-frame that is encoded as a reference to the previous P-frame. These arrows are only examples and do not represent the full set of references of the encoding.

The given "areas" are composed of *macroblocks*. To be more precise, a generic H.264 frame is split into multiple 16×16 squares of pixels, each of them being a macroblock. Macroblocks are encoded/decoded separately from one another, and can be split into sub-blocks down to a block size of 4×4 pixels. Macroblocks are also assigned a type from the set {I, P, B}. I frames can contain only I-blocks. P-frames can contain both P-block and I-blocks. B-frames can contain all types of blocks.

Figure 2 shows an overview of the encoding process. The input frame is divided into macroblocks, each of them is passed to a Coder Control and to a Motion Estimation function. The Motion Estimation function uses some previously encoded and buffered frames, the number of them being determined by the Coder Control. These previous frames are used to choose if the current block should be encoded:

- as a new block, containing the full information (Intra-Frame Prediction, I-block),

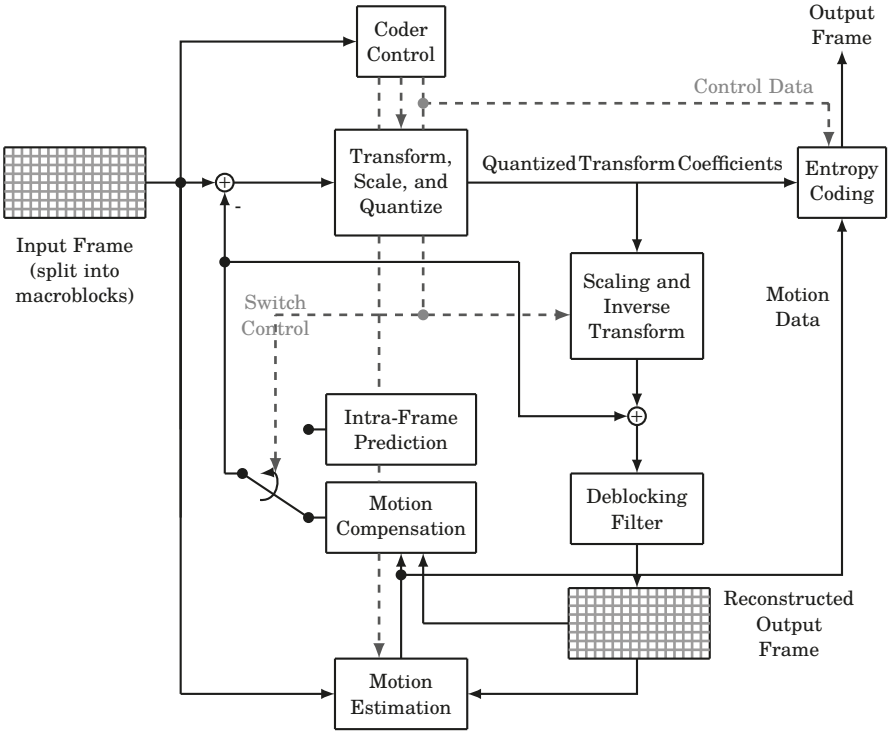


Figure 2. Basic coding structure of a H.264 frame.

- by referring to a previously encoded block in the same frame, containing a positional vector and the residual information (Intra-Frame Prediction, I-block),
- by referring to a block in a previous frame, containing a positional vector, the frame reference, and the residual information (Motion Compensation, P-block), or
- by referring to block in a previous or future frame, containing a positional vector, the frame reference, and the residual information (Motion Compensation, B-block).

The Motion Estimation function determines the cost for the four choices and selects the most appropriate one for the current macroblock.

The residual information is then Transformed, Scaled, and Quantized according to a Quantization Parameter (QP) to reduce its size. This is the only step where there is information loss and the higher the QP value, the higher the loss of information. The scaling, inverse transform and the deblocking filter allow the encoder to reconstruct the output frame and buffer it for future encoding. The entropy coding function uses lossless statistical compression to produce the final output frame.

3. Frame Size Estimation

The aim of this paper is to estimate an upper bound for the size of encoded video frames, to aid a potential external network manager towards a better scheduling of network capacity. The largest improvement is given when information-rich frames (I-frames) are treated separately from frames that can contain references to previous and future frames (P- and B-frames). The small difference in size of P- and B-frames and the similarity in the methods used for their construction justify the use of the same upper bound estimate for the two frame types. We therefore devise two models: an Intra Frame model for I-frames and an Inter Frame model for P- and B-frames. In Section 3.1 we explain what are the implication for network access scheduling. In Section 3.2 we describe the model we use for the estimation of the upper bound of the size of I-frames. In Section 3.3 we describe how to derive upper bounds estimates for P-frames and B-frames. In the following, we use $\propto$ to indicate proportionality.

3.1 Scheduling implications

Assume it is possible to compute an upper bound estimate for the size of I-frames, denoted with I^* and an upper bound estimate for the size of P- and B-frames, denoted with P^*. Knowing the network speed $\mathcal{N}$, e.g., 100 Mbps, one can then translate these bounds into knowledge of the WCTT for the two types of frames in the network. The GOP parameter specifies how many "dynamic" (P- and B-) frames there are in between two "static" (I-) frames.

In fact, when a set $\mathcal{C} = \{c_1, \ldots, c_p\}$ of p surveillance cameras share the same network, one can say that the i-th camera behaves according to the multiframe task model [Mok and Chen, 1997]. The camera has a vector of execution times $[E^0, E^1, \ldots E^{\text{GOP}-1}]$ and a single period and deadline, equal to the inverse of the frame rate $1/f_i$. E^0 is then equal to the upper bound estimate on the transmission time of the I-frame $I^*/\mathcal{N}$ and all the other execution times $[E^1, \ldots E^{\text{GOP}-1}]$ are equal to the upper bound estimates on the transmission time of the P-frame, i.e., $P^*/\mathcal{N}$. This allows us to reuse theoretical results developed for the specific model [Zuhily and Burns, 2009; Han, 1998; Baruah et al., 1999; Lu et al., 2007] or for its generalizations [Baruah et al., 1999; Peng and Fisher, 2016; Stigge et al., 2011; Li et al., 2014; Zeng and Di Natale, 2013; Ekberg et al., 2015; Chakraborty and Thiele, 2005]. In particular, once we have determined the WCTTs for the different frame types, we can use the analysis on non-preemptive scheduling of multiframe tasks [Andersson et al., 2012; Baruah and Chakraborty, 2006] to determine schedulability properties for a set of video-surveillance cameras communicating over switched Ethernet [Andersson, 2008].

As video encoders are very complex software elements, we cannot really compute an upper bound with static analysis or formal methods, that would guarantee that the size will never exceed the one predicted. However, we can compute an approximation (estimate) of such upper bound, that is proven conservative in most cases. We believe that the very few circumstances in which the size of frames exceeds the computed values are due to problems and bugs of the execution of video-surveillance software. Therefore, we refer to $I^*/\mathcal{N}$ and $P^*/\mathcal{N}$ using the term quasi Worst-Case Transmission Times (qWCTT).

3.2 Intra Frame Model – I-frames

To determine the upper bound estimate I^* for the size of I-frames, we isolate the principal components that influence the amount of information included in the frame. Many acronyms and symbols are defined in the rest of the section. Table 2 contains a summary of the terms and constants that are needed for the estimation. The second column of the table contains a letter explaining how the value is obtained: `[C]` for computed, `[K]` for known, `[M]` for measured. Section 3.4 contains details on how to measure the `[M]`-parameters given a scene and a camera model.

Three different components influence the size of the frame: (i) the resolution of the video r, (ii) the compression level I_c, (iii) the actual camera and scene parameters I_a. There are many alternatives to write an expression of how each of these factors influences the size of the resulting frame. We decided to express I_c and I_a as scaling factors with respect to the resolution of the frame, therefore writing I^* as the product of the three terms,

$$I^* = \{r \cdot I_c \cdot I_a\}. \tag{1}$$

We now provide details for each of these terms separately.

- **Resolution r.** The frame resolution r is the number of pixels in the frame. Its value is equal to the product of the height h and the width w of the frame, $r = w \cdot h$. The resolution is linked to the number of macroblocks in the frame, therefore it influences its size directly.

- **Compression level I_c.** We denote with I_c the influence of the compression, $I^* \propto I_c$. The compression level QP determines the loss of information in each macroblock. From the H.264 standard, we infer that "an increase of 1 in QP corresponds to an increase of the quantization step size by approximately 12%" [Wiegand et al., 2003] (an increase of 6 means an increase of the quantization step size by a factor of 2).

  In order to properly capture this relationship, we define a reference QP, denoted with $\mathrm{QP}^{\mathrm{ref}}$, and express I_c as a function of the difference between the current value and the reference value, $\Delta\mathrm{QP} = \mathrm{QP} - \mathrm{QP}^{\mathrm{ref}}$. We select $\mathrm{QP}^{\mathrm{ref}} = 28$ as the baseline. This choice is arbitrary, but represents a commonly used value, and does not affect the generality of the approach. ΔQP is used to scale the frame sizes between two compression levels, according to the relationship $I_c = 2^{-\frac{\Delta\mathrm{QP}}{6}}$. The expression in Equation (1) thus becomes

$$I^* = \{r \cdot I_c \cdot I_a\} = \left\{r \cdot 2^{-\frac{\Delta\mathrm{QP}}{6}} \cdot I_a\right\}. \tag{2}$$

- **Actual camera and scene parameters I_a.** The last component that influences the size of an I-frame includes a mix of camera and scene parameters, that we denote with I_a for "actual". I_a includes two different terms, $I_a = I_d + n_{c,\ell}$. The first one, I_d, is related to how many details the scene has and how well the camera is able to retain that information. The second one, $n_{c,\ell}$ is related to the amount of noise generated in the camera. I^* then becomes

$$I^* = \{r \cdot I_c \cdot I_a\} = \left\{r \cdot 2^{-\frac{\Delta\mathrm{QP}}{6}} \cdot I_a\right\} = \left\{r \cdot 2^{-\frac{\Delta\mathrm{QP}}{6}} \cdot \left(I_d + n_{c,\ell}\right)\right\}. \tag{3}$$

The detail influence I_d, captures how the scene details and their perception at the camera level affect the size of the frame. These can be separated into two categories: (i) scene-dependent parameters (each camera reacts differently to them, but they are a property of the scene), (ii) camera-dependent parameters. Parameters in the first category should be measured, while parameters in the second category are either measured or known, e.g., available from the camera manufacturer.

In the first category, we include the scene illumination ℓ, the scene detail level d_s, and the nature parameter n. In the second category, we include the camera detail level d_c, the enhancing factor e induced by features like High Dynamic Range (HDR), and the Size of the Average Object (SAO) in the scene, which depends for example on the zoom level enforced by the camera. The resulting I_d is the product of all these factors. In fact, the factors are known or measured as the relative difference that they produce in the I-frame size.

The frame size is greatly influenced by the illumination of the surroundings ℓ, given that more light allows the camera to capture the scene better while the absence of light hides details in the image. The value of ℓ represents the ratio between the current illumination level and a reference one, it is is measured in a controlled environment with predetermined light levels. The result of the measurement is a value $\ell \in \mathbb{R}^+ \,|\, 0.25 \leq \ell \leq 1$. We consider three different light levels: low, medium, and high. A low light scenario is a scene recorded at night time, without any major light sources. A medium illumination scene is a night time scenario, with some light source illuminating the scene. A high illumination scene is a daylight scene, or a well lit indoor environment such as an office or a store. The high illumination scenario used as basis for scaling the remaining ones. This means that each camera at high light level has $\ell = 1$, and values for middle and low level are scaling factor that decrease the size of the frame. Given a camera model, these values can be determined experimentally as described in Section 3.4.

Directly connected with the light factor, is the level of details in the scene d_s. The scene detail level represents how many details there are in a scene, and can be measured in the field based on the different scenes. The resulting value is a number $d_s \in \mathbb{R}^+ \,|\, 500 \leq d_s \leq 2000$ expressed in millibits per pixel. Section 3.4 describes how to conduct field measurements.

We have experimentally found that the detail influence is also highly correlated to the amount of nature in the scene—lawns, bushes, trees, and similar. These features increase the difficulty of the encoding process, forcing the encoder to include more details in the resulting image, especially in the presence of wind. A high level description of the scene (e.g., a road, a garden, an office) allows one to provide an estimate of the amount of nature present in the frames. The nature factor n is expressed as the portion of the scene that includes natural elements, $n \in \mathbb{R}^+ \,|\, 0 \leq n \leq 1$. It can be easily measured on the field by taking a frame and computing a rough estimate. Typically, indoor scenes have a nature factor $n = 0$, while forest scenes have a nature factor $n = 1$. Common values for an outdoor parking lot are between 0.5 and 1. The factor included in the computation is $(1 + n)$, as the presence of nature only adds complexity to the scene, compared to the baseline.

The camera properties should be taken into account when computing the detail influence. The factor d_c is used to scale the frame size taking into account factors like

Table 2. Terms and Constants used in the Estimation of the upper bound for the I-frame size.

Acronym or Symbol		Brief Explanation	Range or Typical Values
d_c	[M]	Camera detail level: camera specific constant that reflects the camera capacity to retain scene details	$d_c \in \mathbb{R}^+ \mid 0.1 \leq d_c \leq 10$
d_s	[M]	Scene detail level: indicates the total amount of details in the scene	$d_s \in \mathbb{R}^+ \mid 500 \leq d_s \leq 2000$
e	[M]	Enhancement factor, indicates the effectiveness of High Dynamic Range (HDR) or similar technology	$e \in \mathbb{R}^+ \mid 1 \leq e \leq 1.35$
h	[K]	Height of a frame in pixels	~200–4320
I^*	[C]	Upper bound on the size of I-frames	
I_a	[C]	Influence of camera and scene	
I_c	[C]	Influence of the compression level QP	
I_d	[C]	Influence of the detail level	
ℓ	[M]	Scene illumination: it indicates the luminance (amount of light) in the scene, lower values indicate less light	$\ell \in \mathbb{R}^+ \mid 0.25 \leq \ell \leq 1$
n	[M]	Nature factor: amount of nature (trees, bushes, etc) in the scene	$n \in \mathbb{R}^+ \mid 0 \leq n \leq 1$
$n_{c,\ell}$	[M]	Noise level: camera specific constant indicating the amount of noise in the camera, capturing characteristics like sensor size and type; lower values indicate indoor high light and higher values low-light environments	$n_{c,\ell} \in \mathbb{R}^+ \mid 1 \leq n_{c,\ell} \leq 500$
QP	[K]	Quantization Parameter: reflects the frame compression, higher numbers indicate more information loss	$\mathrm{QP} \in \mathbb{N}^+ \mid 1 \leq \mathrm{QP} \leq 51$
$\mathrm{QP}^{\mathrm{ref}}$	[K]	Reference value used in measurements for the Quantization Parameter QP	28
ΔQP	[K]	$\mathrm{QP} - \mathrm{QP}^{\mathrm{ref}}$	
r	[K]	Frame resolution (number of pixels in the frame)	~64000–35389440
SAO	[M]	Size of Average Object: reflects the expected distance of an object in an image, determined by factors like the zoom level, field of view, and lens type, and placement of the camera	$\mathrm{SAO} \in \mathbb{R}^+ \mid 0.5 \leq \mathrm{SAO} \leq 1.5$
w	[K]	Width of a frame in pixels	~320–8192

the sensor types, lenses properties, etc. The constant value d_c represents how well the camera captures the details in the scene and how sharp they are. A measure of d_c can be obtained with respect to a standard camera. The camera detail level d_c can be measured for a given camera as detailed in Section 3.4.

The dynamic range of the scene, together with the camera's ability of capturing it through various image enhancement techniques such as HDR is modelled using the enhancement factor, e. If one assumes that the different light ranges have the same bitrate characteristics and that the camera auto-exposure will select the range filling the most pixels then $e \in \mathbb{R}^+ \mid 1 \leq e \leq 2$. There are two corner cases, 1 and 2. $e = 1$ describes a scene with no additional dynamic range to capture, such as an indoor scene or a foggy day scene. $e = 2$ describes a scene where half the the frame is low dynamic and the other half is high dynamic, such as an indoor scene with large windows. An average value for all real world scenarios lays in between the two. The cameras that we tested had on average a 35% larger I-frame size when HDR was enabled, inducing $e \in \mathbb{R}^+ \mid 1 \leq e \leq 1.35$.

Another important factor affecting the I-frame size via I_d is the size of typical objects and details in the scene, denoted with the term SAO. This parameter can be approximated based on a combination of the distance to the scene, the zoom level and the field of view. The effect of this is to reduce the I-frame size for scenes where the objects are large, since the amount of details in a typical object usually does not scale with resolution. Section 3.4 provides an explanation of how to estimate this parameter.

The last parameter that we need to include is $n_{c,\ell}$, which captures the influence of noise generated in the camera (which in the end influences the size of the I-frame). We assume that the camera is the only source of noise, but the parameter value varies with the amount of light ℓ. In fact, the amount of noise is in direct relation to the scene noise level. The more light there is, the more sensor saturation, the more photons the sensor receives, and the less noticeable the camera noise becomes. The noise level is heavily camera dependent, and related to both hardware (optics and sensor) and software (exposure strategies, noise filtering technologies, and image settings). Depending on the different light conditions ℓ, the noise level can be measured. Values are $n_{c,\ell} \in \mathbb{R}^+ \mid 1 \leq n_{c,\ell} \leq 500$. The procedure to measure $n_{c,\ell}$ is described in Section 3.4.

Considering all the contributions to the upper bound estimate I^*, and substituting I_d and $n_{c,\ell}$ in Equation (3), we can finally write

$$I^* = \cdots = \left\{ r \cdot 2^{-\frac{\Delta \mathrm{QP}}{6}} \cdot \left(\ell \cdot d_s \cdot (1+n) \cdot d_c \cdot e \cdot \mathrm{SAO} + n_{c,\ell} \right) \right\}, \tag{4}$$

obtaining our desired expression for the I-frame size upper bound estimate.

Figure 3 illustrates the results that we obtain using Equation (4) with a default camera. The figure represents data obtained with three different 1080p videos: v_1, v_2, and v_3. The videos were encoded using different QP values in a standard setup where we know lighting conditions, detail level of both the scene and the camera, the size of objects, the enhancement features and the noise. We record I-frame sizes during the encoding with varying QP values, shown as dots in the Figure. The three lines represent the estimation obtained with Equation (4), which upper bounds the dot in almost every case.

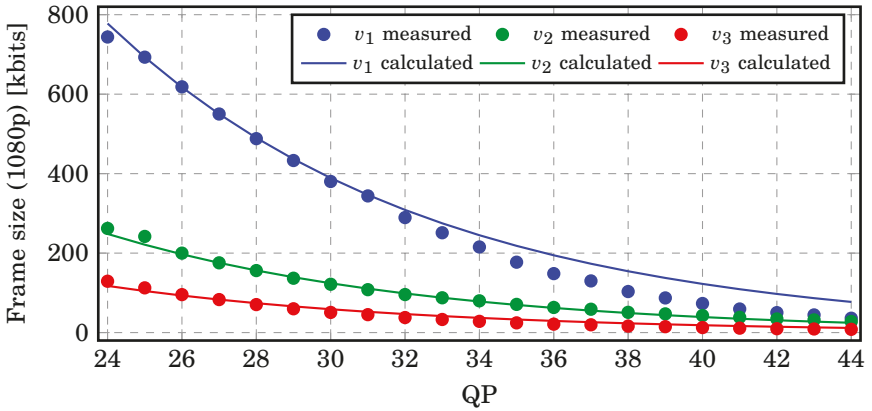


Figure 3. Measured I-frame sizes and calculated ones for different videos, varying QP.

3.3 Inter Frame Model – P-frames and B-frames

The same reasoning we used to estaimte the upper bound of I-frames can be used to estimate the upper bound of the size of frames that can be encoded referencing macroblocks in other frames. The three components that provide contributions to the size of a P- and B-frame are the same. We use P-frames as our basis, as we expect the encoder to be slightly more successful in encoding B-frames, therefore P-frames should represent an upper bound estimates for B-frames too. Table 4 summarizes the additional terms that are defined in this Section.

Using scaling factors with respect to the resolution (as we did for the I-frame), we define

$$P^* = \{r \cdot P_c \cdot P_a\}. \tag{5}$$

The first element contributing to the size of the frame is the resolution of the image r. The second and third components respectively are related to compression (P_c) and to the actual parameters of the camera and scene (P_a).

P-frames are highly correlated with neighboring frames, due to the compression algorithm. This makes the compression factor for P-frames larger than the one for I-frames and $P_c < I_c$. The relation between the compression parameter (QP) and frame size that we used for I-frames does not apply for P-frames due to this correlation. We introduce this by changing the compression term (the base 5 experimentally achieved through curve fitting):

$$P_c = 5^{-\frac{\Delta \mathrm{QP}}{6}}. \tag{6}$$

P_a can again be split into two parts, one part relative to the influence of the detail level P_d and the noise $n_{c,\ell}$, which is the same term used for the I-frames, $P_a = P_d + n_{c,\ell}$. The difference between I_d and P_d, on the contrary, lies in the motion detected in the image. The encoding algorithm tries to find motion, starting from the same macroblock position in buffered images. We therefore encode $P_d = P_m \cdot I_d$, defining P_m as a multiplicative gain that explains the effect of motion on the resulting

Table 4. Additional Terms and Constants used in the upper bound for the P-frame size.

Acronym or Symbol		Brief Explanation	Range or Typical Values
f_{inf}	[K]	Inferior frame rate limit	2
f_{sup}	[K]	Superior frame rate limit	120
f_s^{ref}	[K]	Reference frame rate used for the motion level measurement	30
f_s	[K]	Number of frames per second in the video (saturated)	$f_s \in [f_{\text{inf}}, f_{\text{sup}}]$; ~20–40
P^*	[C]	Upper bound on the size of P-frames	
P_a	[C]	Influence of camera and scene	
P_c	[C]	Influence of the compression level QP	
P_d	[C]	Influence of the detail level	
P_m	[C]	Influence of motion	
μ_s	[C]	Motion level: fraction of the image that is expected to be moving	$\mu_s \in \mathbb{R}^+ \mid 0 \leq \mu_s \leq 1$
μ_x	[M]	Motion encoder efficiency: reflects the ability of efficiently encode moving object, an encoder with a large motion search window will have a low motion cost	$\mu_x \in \mathbb{R}^+ \mid 0 \leq \mu_x \leq 1$

frame size, refining Equation (5) into

$$P^* = \{r \cdot P_c \cdot P_a\} = \left\{ r \cdot 5^{-\frac{\Delta \text{QP}}{6}} \cdot \left(P_m \cdot I_d + n_{c,\ell}\right) \right\}. \tag{7}$$

The influence of motion on the P-frame size P_m is affected by three factors: (i) the frame rate f_s, (ii) the scene motion level μ_s, and (iii) a camera motion cost, which reflects how well the H.264 encoder captures encoding of moving objects, which we call motion encoder efficiency μ_x.

- **Saturated frame rate f_s:** P_m is directly linked to the frame rate of the video: the lower the frame rate, the more difference there will be between consecutive frames, the larger the motion step will be and the more objects would have moved. This larger gap will translates into higher chances of a motion miss by the encoder, and leads to higher bandwidth consumption. At extremely high frequencies or extremely low frequencies, the frame rate effect saturates. We therefore impose thresholds on the frame rate, forcing it to belong to the interval $[f_{\text{inf}}, f_{\text{sup}}]$. We have experimentally determined good values for f_{inf} and f_{sup} and respectively set them to 2 and 120. Using experimental data, we have determined that P_m is proportional to the inverse square of the video frame rate $P_m \propto \sqrt{f_s}^{-1}$.
- **Scene motion level μ_s:** The motion level of a scene is a measurable quantity at a certain reference frame rate f_s^{ref}, in our case equal to 30. This means that $P_m \propto \mu_s \cdot \sqrt{f_s^{\text{ref}}}$. The motion level determines the portion of the image that has

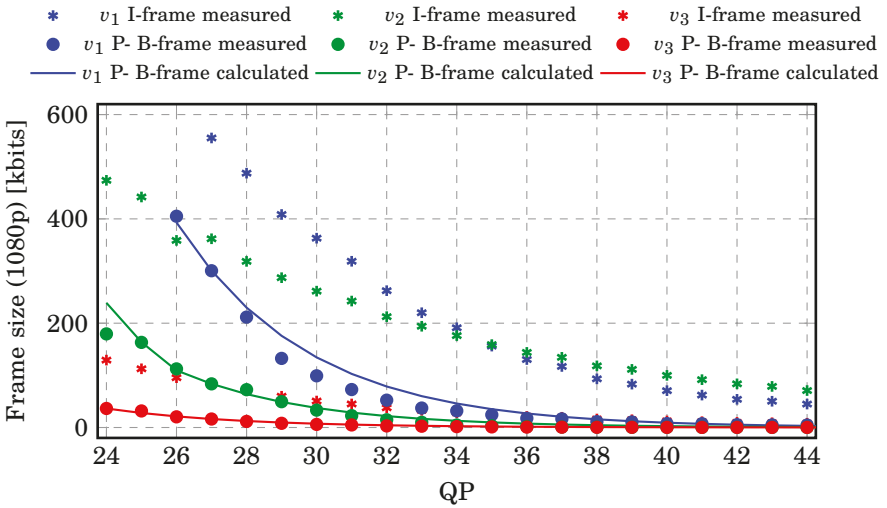


Figure 4. Measured P- and B-frame sizes and calculated ones for different videos, varying QP.

moved from one frame to the next. If accurately known, μ_s can be uniquely used and varied per frame. However, since the primary use case of our upper bound is to estimate the required bandwidth there is a strong added benefit in simplifying the analysis. For simplification, we only use a generic set of possible motion levels: high, medium, and low. For high motion scenes, μ_s is typically around 0.15. For medium motion scenes, its value is around 0.07, and for low motion scenes 0.01.

- **Motion encoder efficiency μ_x:** The motion encoder efficiency is a measurable quantity per camera. The camera encoding capabilities are often dependent on the encoder capabilities and efficiency. The motion encoder efficiency can be measured, as explained in Section 3.4.

Including all the terms specified above, one can write $P_m = \mu_s \cdot \mu_x \cdot \sqrt{f_s^{\text{ref}}/f_s}$, and therefore, substituting P_m in Equation (7), we obtain our upper bound estimate

$$P^* = \cdots = \left\{ r \cdot 5^{-\frac{\Delta \text{QP}}{6}} \cdot \left(\mu_s \cdot \mu_x \cdot \sqrt{f_s^{\text{ref}}/f_s} \cdot I_d + n_{c,\ell} \right) \right\}. \tag{8}$$

Figure 4 illustrates the results that we obtain using Equation (8) with a default camera with known parameters. The figure represents data obtained with three different 1080p videos, v_1, v_2, and v_3. The lines represent the estimation obtained with Equation (8), which upper bounds the measured values, plotted with dots. We also report the measured size of I-frames for the same videos with asterisks.

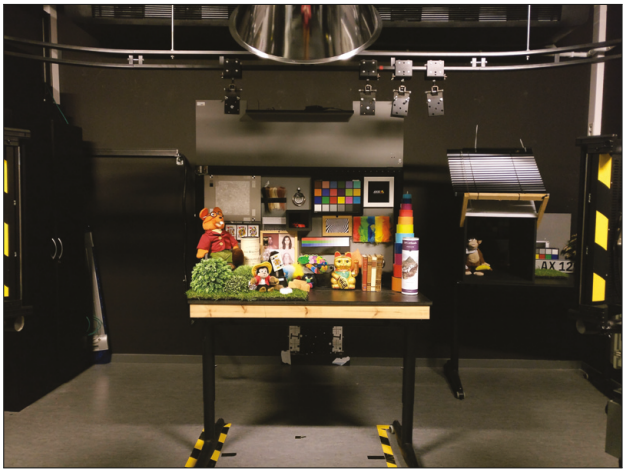


Figure 5. Image laboratory used to determine characteristics related to the camera and the scene.

3.4 Model Calibration

As indicated above, different constants need to be measured for the various cameras and scenes, in order to be able to extract meaningful numbers for Equations (4) and (8). These characteristics can be grouped in different sets: (i) platform-related, (ii) camera-related, and (iii) scene-related.

- **Platform-related characteristics.** The motion encoder efficiency μ_x is related to the platform (mostly the encoder) that is being used. In principle, the scene is also important in this case, but a scene-independent approximation can be computed. For each encoder generation and brand, the estimation of μ_x is done by isolating the encoder, or an equivalent encoder model, with a series of predetermined video sequences, encoded using varying compression.
- **Camera-related characteristics.** The three characteristics that we need to measure among the camera-related ones are d_c, $n_{c,\ell}$, and e. They are respectively: (d_c) the ability of the camera to retain scene details, ($n_{c,\ell}$) the amount of noise that the camera generates in specific light conditions, and (e) the enhancement factor added by technology like HDR. These are constants that summarize many different physical elements like the sensor size and quality.
- **Scene-related characteristics.** Four scene-related characteristics should be measured: the scene level detail d_s, the amount of nature n, the Size of the Average Object in the scene, SAO, and the amount of light ℓ.

Measurements should be collected in a reproducible environment. In our case, we collected the data in a dedicated laboratory. The main idea is to be able to reproduce certain scene conditions. The environment must contain different levels of details. It should be possible to shoot videos of areas with few or no details, as well as others with many details. It should also be possible to control the amount of light, at least

to reproduce three different light conditions — high, medium, and low. Finally, there should be some reproducible source of motion, e.g., a fan or a toy train. The position of the camera with respect to the scene should be fixed in advance and should be reproducible as well. Figure 5 shows the laboratory in which the tests to compute the above mentioned parameters were conducted. Most measurements are conducted using a reference camera, and then for a new camera some additional data is collected to compare the camera to the reference one.

To determine the parameters we follow a specific procedure, both for the reference camera and for the model that we are trying to profile: (i) we record (repeatable) scenes with no motion, motion, no details, details, in three different light levels; using the compression level $\mathrm{QP}^{\mathrm{ref}}$; (ii) we extract the frame sizes for all the I-frames and P-frames in the video; and (iii) we compute statistics for the videos, the average and maximum size of I- and P-frames.

For the camera detail level d_c, we compute the average frame size (for all the set of recorded videos), including both I-frames and P-frames and compare them with the values obtained with the reference camera. Denoting with S_{avg} the average frame size of the camera under test and with $S^{\mathrm{ref}}_{\mathrm{avg}}$ the one of the reference camera, $d_c = S_{\mathrm{avg}}/S^{\mathrm{ref}}_{\mathrm{avg}}$. We repeat the same considering only low light conditions, and compute $n_{c,\ell}$ exactly using the same formula. The value of e for a given camera is determined by computing the ratio of the average frame sizes with HDR activated and deactivated.

To compute scene-level measurements, there are two alternatives. The first one is to physically record videos from the location where the camera should be installed, and the second one is to film similar scenes multiple times, and re-use the average measured parameter for similar scene types. We denote with $S_{I,\mathrm{avg}}$ the average size of I-frames measured in bits for these measurements. We also want to collect videos done with the zoom level set to 50% for this case. The average size of the I-frames for this zoom level is indicated with $S_{I,50\%,\mathrm{avg}}$.

The reference camera is used to measure the scene detail level d_s. Using the set of videos recorded from similar or the same scene, d_s is computed as the average size of I-frames expressed in millibits per pixel, $d_s = S_{I,\mathrm{avg}} \cdot 1000/r$. The scene illumination ℓ is measured by comparing the laboratory result with the scene results using the actual camera to be used. From the laboratory results, we take videos recorded in high illumination scenes and compute the average size of I-frames for these videos as $S_{I,\ell=1,\mathrm{avg}}$. We then compute ℓ as $\ell = S_{I,\ell=1,\mathrm{avg}}/S_{I,\mathrm{avg}}$. The amount of nature n is computed by looking at how many pixels in a frame are covered by nature.

Finally, we need to measure the size of the average object SAO. SAO is determined as $S_{I,50\%,\mathrm{avg}}/S_{I,\mathrm{avg}}$. The SAO levels can be, for simplicity, divided into three levels: large, medium, and small. As a general rule of thumb, one can determine the SAO level for 1080p video such as: (i) Large SAO: Objects taking up more than 1% of the pixels. An example is a licence plate camera, commonly setup to capture mainly a car with sufficient margin around it. (ii) Medium SAO: Objects are between 1% and 0.01% of the pixels. This is the most common case. (iii) Small SAO: Objects are very small, less than 0.01% of the pixels. This is sufficient only for scene awareness, i.e. knowing what happened in the scene, but does not permit to identify objects.

4. Related Work

The ultimate goal of this paper is to enable scheduling of network bandwidth in a video-surveillance system, utilizing the available bandwidth as much as possible. This goal can be achieved in many different ways.

One alternative to better utilize network resources is to reduce the amount of sent information by exploiting better compression and enhanced encoding. A lot of research has been devoted to adapting video stream quality to fit network channels [Rinner and Wolf, 2008; Ramos et al., 2007; Seetanadi et al., 2017; Kotra and Fohler, 2010; Kotra and Fohler, 2011; Almeida et al., 2007]. For example, adaptive strategies have been developed for MJPEG encoding [Seetanadi et al., 2017; Almeida et al., 2007], MPEG-2 [Kotra and Fohler, 2010], and MPEG-4 [Kotra and Fohler, 2011]. Another alternative offer variable network channels [Wang et al., 2008; Seetanadi et al., 2017]. In this work, we investigate estimation of the WCTT for frames over a network, which is related to these works, but takes a different route. The aim of this paper is to devise a reasonably accurate model to aid scheduling decisions, without introducing adaptation.

To the best of our knowledge, there are two known alternative methods to estimate the frame size, and in turn the expected video bandwidth needed for the video transmission. These methods are based on other encoding methods (respectively MJPEG and MPEG-4) and aim to provide an estimate of the expected frame sizes. To the best of our knowledge, we propose the first frame size estimation for MPEG-4 part 10 AVC (H.264).

We denote the MJPEG method with `LIN`. This method only considers the compression parameter (QP for H.264 videos), and scales the frame size linearly according to such a parameter that we name q_l. Given a maximum size, identified with the term s_{max}, the frame size $s(q_l)$ is computed as $s(q_l) = q_l \cdot s_{max}$. The parameter q_l indicates the quality of the encoding, and relates, as indicated previously, to the Quantization Parameter QP. The scale and logic used are different and in MJPEG $q_l \in [0.01, 1.0]$, 1 being the lowest compression and 0.1 the highest, therefore $q_l = 1.01 - (\mathrm{QP}/51)$. In the case of a 1080p YCbCr color video with 8 bits per pixel, $s_{max} = 1920 \cdot 1080 \cdot 8 \cdot 3 = 49766400$ [bits per frame]. This model is used for example in [Seetanadi et al., 2017; Seetanadi et al., 2017] to devise a control strategy to determine the quality to be applied given a target bandwidth consumption.

We call the MPEG-4 model `RQM`. This model is used in [Almeida et al., 2007] and described in [Ding and Liu, 1996]. It uses curve fitting to determine the parameters of a rate-distortion curve, modeled with a Gaussian random variable. Denoting with α a constant accounting for overhead bits, with β a constant that varies with the resolution and amount of motion in the video, with q_r the compression level for MPEG-4 ($q_r \in [1, 31]$), and with γ a constant that varies depending on the frame type (paper [Ding and Liu, 1996] providing recommended bounds of $\gamma \in [0.5, 1]$ for I-frames and $\gamma \in [0.5, 1.5]$ for P-frames), the size of the frame can be written as $s(q_r) = \alpha + \beta \cdot 1/{q_r}^{\gamma}$.

Notice that neither `LIN`, nor `RQM` compute proper upper bounds. They rather compute estimates of the frame size. We therefore do not expect them to be suitable for upper bounding the size of frames and obtaining WCTTs.

Table 5. Measured camera-related parameters.

Model	d_c	$n_{c,l}$ (high ℓ)	$n_{c,l}$ (medium ℓ)	$n_{c,l}$ (low ℓ)	μ_x
A	1.00	2.50	2.75	22.2	0.450
B	0.98	0.25	2.75	230.0	0.450
C	1.23	0.35	1.10	102.0	0.450
D	0.54	0.75	4.05	5.6	0.400
E	0.81	1.25	12.00	35.0	0.400
F	1.03	2.25	2.7	119.0	0.425

5. Experimental Results

In this section we present our experimental evaluation. We conducted many tests with different cameras and in different scenarios to validate the upper bounds estimates computed with our technique. We present two different categories of tests. Section 5.1 shows the results obtained for a controlled environment and a repeatable video, comparing our estimation strategy with state-of-the-art techniques. Section 5.2 presents a stress-test where we report the aggregate results of a large experimental campaign.

To conduct a comprehensive evaluation, we used 6 different camera models, and deployed them in 24 real-life (surveillance) scenarios. We refer to the different camera models using letters from A to F. Camera A was used as reference camera for the parameter estimation discussed in Section 3.4. To show the versatility of the model we use different parameters, resolutions, etc. Also, Camera C is a thermal camera. Table 5 contains the camera-related parameters that do not change with the scenario. Parameters that change with the scenario will be discussed in the corresponding sections.

5.1 Frame-by-Frame Evaluation

We present here a first validation experiment done with our reference Camera A. We recorded two videos of the same scene in our laboratory. The scene has a lot of details. The laboratory allows us to move the camera with predictable motion and control the amount of movement introduced in the image. Our aim is to show a frame-by-frame comparison between our frame size estimation and the state-of-the-art techniques discussed in Section 4.

The two videos differ in the amount of motion that is introduced[3]. A toy, present in the scene, allows us to introduce very limited but non-zero motion in both cases. In the first video, we also sharply changed the position of the camera. This simulates a fast movement for a video-surveillance camera. In the second video we kept the camera still, thus the only movement comes from the toy. The first video is characterized by a large amount of motion μ_s, while the second video has a very low μ_s.

[3] The two videos are available online: https://www.youtube.com/watch?v=614BbbhD56M (high-motion), and https://www.youtube.com/watch?v=q4j3LlVr0ls (low-motion). We have manipulated them to also visually show the motion vectors detected for both the original videos: https://www.youtube.com/watch?v=5YrxlGhadsY (high-motion), and https://www.youtube.com/watch?v=cfrO8CZQa-E (low-motion)

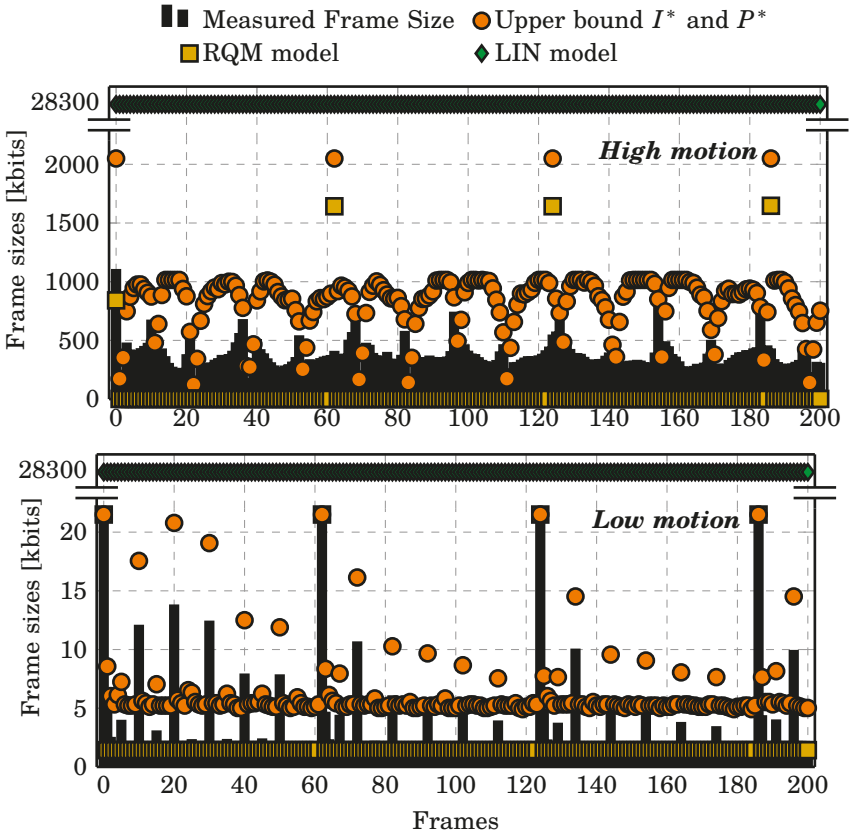


Figure 6. Results of the comparison experiment with the high- and low-motion video.

The Camera A parameters for the two videos are: camera level detail $d_c = 1$, enhancement factor $e = 1.35$ (HDR), width $w = 1920$[pixels], height $h = 1080$[pixels], frame rate $f_s = 25$[frames per second], QP = 29, noise level $n_{c,\ell} = 2.5$, motion encoder efficiency $\mu_x = 0.45$, GOP = 64. The scene parameters are: no nature, $n = 0$, very good illumination, $\ell = 1$, scene detail $d_s = 780$[millibit per pixel], and size of the average object SAO = 1.

Figure 6 shows the results we obtained for the two videos. Each plot represents 200 frames of one video, the top one being the high-motion one and the bottom one being the low-motion case. The black bars represent the real frame sizes measured after the encoding. The circles represent the estimated upper bound on the frame sizes provided by the algorithm presented in this paper. The squares show the estimate produced by the LIN model, which does not take into account the difference between I, P, and B frames. Finally, the squares represent the estimate produced by the RQM model.

For the RQM model, we used to the low-motion video to tune the parameters α, β, and γ, as recommended in [Almeida et al., 2007]. The tuning resulted in $\alpha = 0.55$ and $\beta = 1.7$. As γ changes depending on the frame type, we fit $\gamma_I = 0.5$ and $\gamma_P = 4$ separately. The RQM tuning resulted in average errors on I-frames and P-frames respectively of 1.80% and 1.38%, which indicate very good performance for the low motion video. The square points in the lower plot of Figure 6 are therefore *a posteriori* estimations, and are clearly a very good fit for the video, despite the presence of a few outliers. The RQM model neglects motion — i.e., the β parameter is not sufficient to take motion into account. In fact, when the parameters determined with the low-motion video are used for *a priori* estimating the size of the frames in the high-motion video, the estimate frame size greatly underestimates the real value. The RQM approximation is therefore not a good fit to upper bound the size of the frames.

On the contrary, the LIN model gives very conservative results for both the high- and low-motion video, as its only parameter is a translation of the encoding quality QP. These are too conservative to be used in any practical setting, since the estimates are roughly 30 times as large as the real values. The LIN approximation is therefore also not a good upper bound for the size of the frames.

In the case of our upper bound estimates I^* and P^*, the circles represent for both plots *a priori* estimates based on the parameters that we have selected and on a standard computation of the motion level μ_s based on the percentage of pixels that differ from one image to the next (which could be determined before the encoding step). Roughly, the computed upper bound estimates are twice as large as the real values. While this could be reduced with a more conservative setup of parameters, we believe that there could be a risk of cases in which the real frame size exceeds the upper bound estimate. In the full length of the two videos (low-motion 751 frames, high-motion 376 frames) this never happens for the low-motion case, and happens five times for the high-motion case. Inspecting these five occurrences prompted us to suspect some capturing error or some encoding miss, possibly due to the sharp movement.

5.2 Stress test

The purpose of the stress test is to verify that we obtain a reasonably good estimate of the bandwidth consumed by cameras to transmit their frame streams to a base station. We deployed our cameras in real-life surveillance scenarios and collected video streams for a time up to five days. We then measured the expected bandwidth consumption using estimates of the parameters (e.g., instead of computing precisely the motion level μ_s, we guessed it based on the type of recorded scene). We compared the measure expected bandwidth with the real bandwidth requirements — the videos' bitrates. The characteristics of the tested scenarios and the obtained results are summarized in Table 6, where b_r represents the bitrate, and $\hat{b_r}$ its estimate.

The scene in scenarios 1a–1f is a highly illuminated parking lot, recorded with camera A ($e = 1.35, w = 1920, h = 1080$). Scenarios 2a–2k are videos from the surveillance system of a hotel complex. Camera B ($w = 1920, h = 1080$) in scenario 2a points at the reception entrance. In scenario 2b, Camera B ($w = 1920, h = 1080$) captures the emergency exit. Camera C ($w = 1920, h = 1080$) in scenario 2c films the control room. Camera A ($w = 1920, h = 1080$) in 2d is directed to the parking entrance. Camera C

Table 6. Parameters and results of the experiments conducted with 6 cameras in 24 real-life surveillance scenarios.

	f_s	QP	GOP	μ_s	ℓ	d_s	SAO	b_r	$\hat{b_r}$
1a (A)	25	28	62	$\approx 1\%$	1	780	1.00	1040	1275
1b (A)	25	28	62	$\approx 3\%$	1	780	1.00	1600	1806
1c (A)	25	28	62	$\approx 9\%$	1	780	1.00	3200	3398
1d (A)	12	32	32	$\approx 1\%$	1	780	1.00	544	600
1e (A)	12	32	32	$\approx 3\%$	1	780	1.00	720	723
1f (A)	12	32	32	$\approx 11\%$	1	780	1.00	1200	1219
2a (B)	15	28	62	$\approx \{2,3\}\%$	{1, 0.8}	810	1.00	794	991
2b (B)	15	28	62	$\approx 0\%$	1	710	0.45	78	208
2c (C)	15	28	62	$\approx 1\%$	0.8	820	0.45	243	287
2d (A)	15	28	62	$\approx \{3,5\}\%$	{1, 0.8}	990	0.45	669	765
2e (C)	15	28	62	$\approx 1\%$	1	810	1.00	513	761
2f (C)	15	28	62	$\approx 1\%$	{1, 0.8}	1400	1.00	333	490
2g (C)	15	28	62	$\approx 5\%$	1	920	0.45	409	456
2h (F)	15	28	62	$\approx 0\%$	0.8	710	0.45	45	96
2i (A)	15	28	62	$\approx 0\%$	{1, 0.5}	780	1.10	722	793
2j (F)	15	28	62	$\approx 4\%$	0.8	780	1.00	139	144
2k (A)	15	28	62	$\approx \{4,3\}\%$	{1, 0.5}	780	1.00	194	220
3a (A)	25	28	32	$\approx 21\%$	1	1200	1.00	10000	10051
3b (A)	25	28	32	$\approx 4\%$	1	1200	1.00	2800	3116
4a (C)	30	18	32	$\approx 6\%$	1	660	1.00	4215	4551
4b (C)	30	18	32	$\approx 2\%$	0.5	780	1.00	4966	5321
5 (D)	25	24	4	$\approx 2\%$	1	990	1.00	42500	46529
6 (E)	25	32	32	$\approx 4\%$	0.5	660	1.00	2837	2878
7 (A)	15	36	30	$\approx 20\%$	1	1050	1.00	620	681

($w = 1920, h = 1080$) in 2e films the reception. Camera C ($w = 1280, h = 720$) in 2f captures the corridor with shops. In 2g, Camera C ($w = 1280, h = 720$) is directed towards the elevator. Camera F in 2h films the staircase. Camera A ($w = 1280, h = 720$) in 2i streams a parking lot with nature $n = 0.5$. Camera F ($w = 704, h = 480$) in 2j and Camera A ($w = 704, h = 480$) in 2k film parking lots without nature. When the table contains two numbers for the motion level μ_s and for the light ℓ, this means that in the estimation the numbers are adjusted for day and night capture. The set includes first the day and then the night value. The value of e is set to 1 for 2b, 2c, 2e, 2f, 2h, 2j, which means HDR is turned off. In the other scenarios, HDR is turned on with a contribution of $e = 1.35$. The two instances of Camera A ($e = 1.35, w = 1920, h = 1080$) used in scenario 3a and 3b are placed in bridges on the highway and monitor car traffic. The two instances of Camera C ($e = 1$) of scenario 4a and 4b monitor a perimeter of a parking lot and the parking lot itself. In 4a the resolution is set to $w = 640, h = 480$, while in 4b the resolution is set to $w = 384, h = 288$. In scenario 5, Camera D ($e = 1, w = 3840, h = 2160$) streams a 4k video of the corner of a city street. Camera E ($e = 1.35, w = 3072, h = 1728$) in scenario 6 is filming a shipyard loading dock. Finally, in scenario 7 Camera A ($e = 1.35, w = 1280, h = 720$) is facing a city intersection.

Despite the high variety of scenes, the varying light conditions, the different cam-

eras, and the different motion levels, the estimated bitrate $\hat{b_r}$ (upper bound estimate) is always higher than the measured bitrate b_r. In most cases, the two values are very similar to one another (see for example scenario 1e or 3a). In a few cases, like 2b and 2h, it is possible to see that the upper bound overestimates the video bitrate (respectively 2.65 and 2.13 times as large). However, we believe these numbers provide a reasonable upper bound estimate and permit to correctly dimension the network bandwidth, aiding scheduling decisions.

6. Conclusion and Future Work

In this paper we presented a practical contribution on how to derive upper bounds estimates for the size of video frames in a streaming system. We have discussed which characteristics influence the bandwidth requirements of different cameras, derived models for the upper bound estimates of the size of I-, P-, and B-frames. We have also systematized the knowledge on the involved quantities and parameters. We divided such quantities into parameters that are known, characteristics that are measurable, and values that are computable. We have then taken the measurable characteristics and discussed how to conduct field tests to obtain reasonable values for them, and — when possible — how to guess based on the environmental conditions. Some parameters can be more or less easily estimated online (motion, light level, noise level, scene type...). Estimating these parameters on the source could lead to a more accurate and less pessimistic short term prediction. More frame by frame tests as well as highly challenging scenarios will also be ran in order to enhance the model.

The derivation of reasonable upper bounds estimates for the WCTT allows us to precisely formulate the problem of allocating network bandwidth to a set of cameras in a switched Ethernet network environment and to reuse well-known scheduling results. We have shown with a thorough experimental campaign that our estimated upper bounds are more reliable, and closer to the real frame sizes than state-of-the-art estimation techniques.

A proper estimation of the frame sizes is the key to properly dimension network infrastructures for real-time video-surveillance systems. Our results demonstrated that we can dimension the network infrastructure, being able to accurately predict the bitrate consumption of video streams. Our findings have a significant industrial relevance, as they permit to reduce the infrastructure cost and allows us to reuse known scheduling results.

References

Almeida, L., P. Pedreiras, J. Ferreira, M. Calha, J. A. Fonseca, R. Marau, V. Silva, and E. Martins (2007). “Online QoS adaptation with the flexible time-triggered (FTT) communication paradigm”. In: *Handbook of Real-Time and Embedded Systems*.

Andersson, B. (2008). “Schedulability analysis of generalized multiframe traffic on multihop-networks comprising software-implemented ethernet-switches”. In: *IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–8. DOI: 10.1109/IPDPS.2008.4536565.

Andersson, B., S. Chaki, D. de Niz, B. Dougherty, R. Kegley, and J. White (2012). “Non-preemptive scheduling with history-dependent execution time”. In: *24th Euromicro Conference on Real-Time Systems*, pp. 363–372. DOI: 10.1109/ECRTS.2012.38.

Baruah, S., D. Chen, S. Gorinsky, and A. Mok (1999a). “Generalized multiframe tasks”. *Real-Time Systems* **17**:1, pp. 5–22. ISSN: 0922-6443. DOI: 10.1023/A:1008030427220. URL: https://doi.org/10.1023/A:1008030427220.

Baruah, S., D. Chen, and A. Mok (1999b). “Static-priority scheduling of multiframe tasks”. In: *Real-Time Systems, 1999. Proceedings of the 11th Euromicro Conference on*, pp. 38–45. DOI: 10.1109/EMRTS.1999.777448.

Baruah, S. K. and S. Chakraborty (2006). “Schedulability analysis of non-preemptive recurring real-time tasks”. In: *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*. DOI: 10.1109/IPDPS.2006.1639406.

Chakraborty, S. and L. Thiele (2005). “A new task model for streaming applications and its schedulability analysis”. In: *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*. DATE ’05. IEEE Computer Society, Washington, DC, USA, pp. 486–491. ISBN: 0-7695-2288-2. DOI: 10.1109/DATE.2005.26.

Ding, W. and B. Liu (1996). “Rate control of mpeg video coding and recording by rate-quantization modeling”. *IEEE transactions on circuits and systems for video technology* **6**:1, pp. 12–20.

Ekberg, P., N. Guan, M. Stigge, and W. Yi (2015). “An optimal resource sharing protocol for generalized multiframe tasks”. *Journal of Logical and Algebraic Methods in Programming* **84**:1, pp. 92–105. ISSN: 2352-2208. DOI: https://doi.org/10.1016/j.jlamp.2014.10.001.

Han, C.-C. J. (1998). “A better polynomial-time schedulability test for real-time multiframe tasks”. In: *Proceedings 19th IEEE Real-Time Systems Symposium*, pp. 104–113. DOI: 10.1109/REAL.1998.739735.

ISO/IEC MPEG & ITU-T VCEG, J. V. T. (of (2003). *”Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14496-10 AVC)”*.

ITU-T (2017). *Advanced video coding for generic audiovisual services*. https://www.itu.int/rec/T-REC-H.264-201704-I/en. (Visited on 2017-08-23).

Kotra, A. and G. Fohler (2010). "Resource aware real-time stream adaptation for mpeg-2 transport streams in constrained bandwidth networks". In: *IEEE International Conference on Multimedia and Expo*, pp. 729–730. DOI: 10.1109/ICME.2010.5583196.

Kotra, A. and G. Fohler (2011). "Resource aware real-time stream adaptation of mpeg-4 video in constrained bandwidth networks". In: *Visual Communications and Image Processing*, pp. 1–4. DOI: 10.1109/VCIP.2011.6116008.

Li, S., S. Rubini, F. Singhoff, and M. Bourdelles (2014). "A task model for tdma communications". In: *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, pp. 1–4. DOI: 10.1109/SIES.2014.7087455.

Lu, W.-C., K.-J. Lin, H.-W. Wei, and W.-K. Shih (2007). "New schedulability conditions for real-time multiframe tasks". In: *19th Euromicro Conference on Real-Time Systems (ECRTS'07)*, pp. 39–50. DOI: 10.1109/ECRTS.2007.20.

Mok, A. K. and D. Chen (1997). "A multiframe model for real-time tasks". *IEEE Transactions on Software Engineering* **23**:10, pp. 635–645. ISSN: 0098-5589. DOI: 10.1109/32.637146.

Pedreiras, P. and L. Almeida (2003). "The flexible time-triggered (FTT) paradigm: an approach to QoS management in distributed real-time systems". In: *International Parallel and Distributed Processing Symposium*.

Peng, B. and N. Fisher (2016). "Parameter adaption for generalized multiframe tasks and applications to self-suspending tasks". In: *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 49–58. DOI: 10.1109/RTCSA.2016.15.

Ramos, N., D. Panigrahi, and S. Dey (2007). "Dynamic adaptation policies to improve quality of service of real-time multimedia applications in IEEE 802.11e WLAN networks". *Wireless Networks* **13**:4, pp. 511–535. ISSN: 1022-0038. DOI: 10.1007/s11276-006-9203-5.

Rinner, B. and W. Wolf (2008). "An introduction to distributed smart cameras". *Proceedings of the IEEE* **96**:10.

Seetanadi, G. N., J. Cámara, L. Almeida, K.-E. Årzén, and M. Maggio (2017a). "Event-driven bandwidth allocation with formal guarantees for camera networks". In: *IEEE Real-Time Systems Symposium*.

Seetanadi, G. N., L. Oliveira, L. Almeida, K.-E. Arzen, and M. Maggio (2017b). " Game-theoretic network bandwidth distribution for self-adaptive cameras". In: *15th International Workshop on Real-Time Networks*.

Stigge, M., P. Ekberg, N. Guan, and W. Yi (2011). "The digraph real-time task model". In: *Proceedings of the 2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*. RTAS '11. IEEE Computer Society, Washington, DC, USA, pp. 71–80. ISBN: 978-0-7695-4344-4. DOI: 10.1109/RTAS.2011.15. URL: http://dx.doi.org/10.1109/RTAS.2011.15.

Tang, L., Q. Huang, A. Puntambekar, Y. Vigfusson, W. Lloyd, and K. Li (2017). "Popularity prediction of facebook videos for higher quality streaming". In: *USENIX Annual Technical Conference, USENIX ATC*.

Toka, L., A. Lajtha, É. Hosszu, B. Formanek, D. Géhberger, and J. Tapolcai (2017). "A Resource-Aware and Time-Critical IoT framework". In: *IEEE International Conference on Computer Communications*.

Vandalore, B., W.-C. Feng, R. Jain, and S. Fahmy (2001). "A survey of application layer techniques for adaptive streaming of multimedia". *Real-Time Imaging* **7**:3.

Veeraraghavan, V. and S. Weber (2008). "Fundamental tradeoffs in distributed algorithms for rate adaptive multimedia streams". *Computer Networks* **52**:6.

Wang, X., M. Chen, H.-M. Huang, V. Subramonian, C. Lu, and C. D. Gill (2008). "Control-based adaptive middleware for real-time image transmission over bandwidth-constrained networks". *IEEE Transactions on Parallel and Distributed Systems* **19**:6.

Wiegand, T., G. J. Sullivan, G. Bjontegaard, and A. Luthra (2003). "Overview of the H.264/AVC video coding standard". *IEEE Transactions on Circuits and Systems for Video Technology* **13**:7, pp. 560–576. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2003.815165.

Zeng, H. and M. Di Natale (2013). "Outstanding paper award: using max-plus algebra to improve the analysis of non-cyclic task models". In: *2013 25th Euromicro Conference on Real-Time Systems*, pp. 205–214. DOI: 10.1109/ECRTS.2013.30.

Zuhily, A. and A. Burns (2009). "Exact scheduling analysis of non-accumulatively monotonic multiframe tasks". *Real-Time Systems* **43**:2, pp. 119–146. ISSN: 0922-6443. DOI: 10.1007/s11241-009-9085-6. URL: http://dx.doi.org/10.1007/s11241-009-9085-6.

Paper II

Control-Based Resource Management for Storage of Video Streams

Alexandre Martins Mikael Lindberg Martina Maggio, Karl-Erik Årzén

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.

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

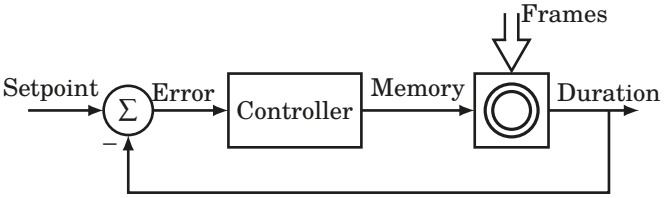


Figure 1. Video storage feedback loop.

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, 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.

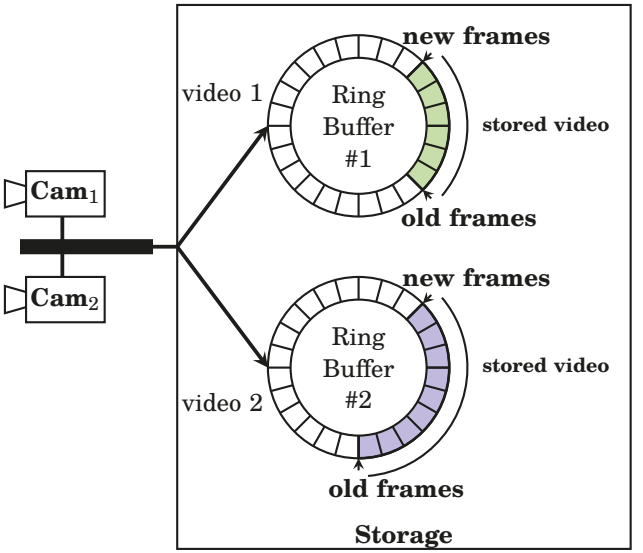


Figure 2. Multiple video streams sharing the same disk space.

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 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.

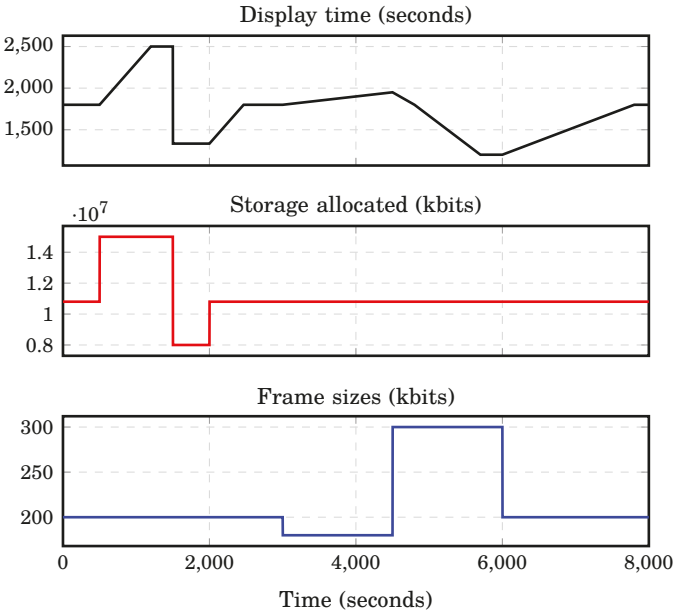


Figure 3. Stored video time, allocated storage, and frame size in open loop.

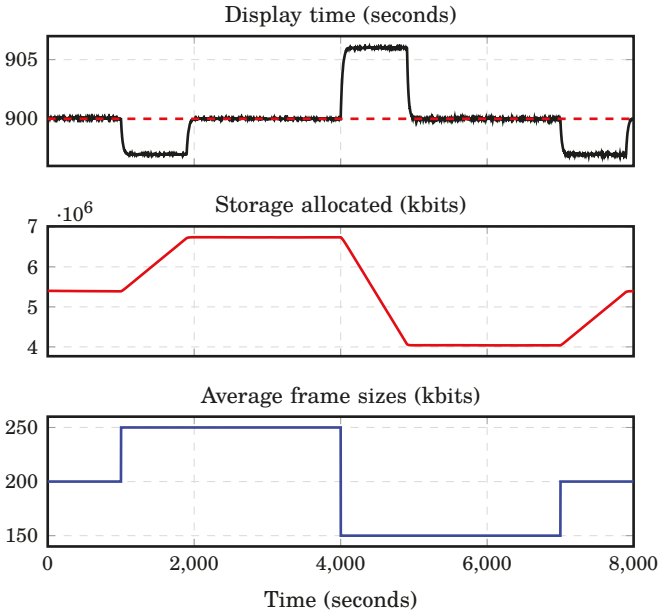


Figure 4. Average frame size, target storage and buffer behavior in closed loop (K=5000, Ti=0.002).

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]

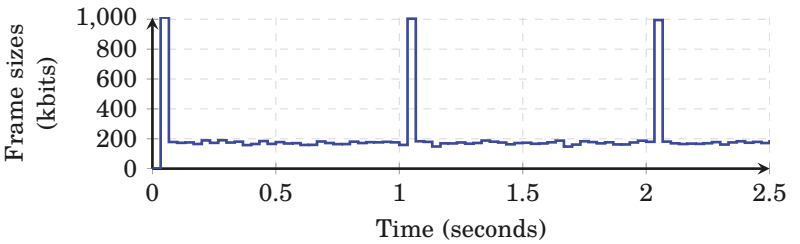


Figure 5. Frame size detail in closed loop.

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 ([Åström 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

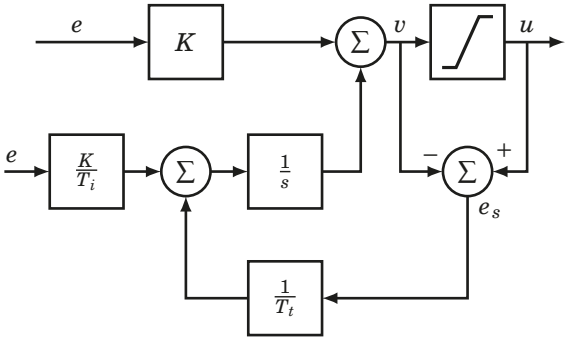


Figure 6. A PI controller with tracking-based anti-windup.

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;
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*.

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.

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 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

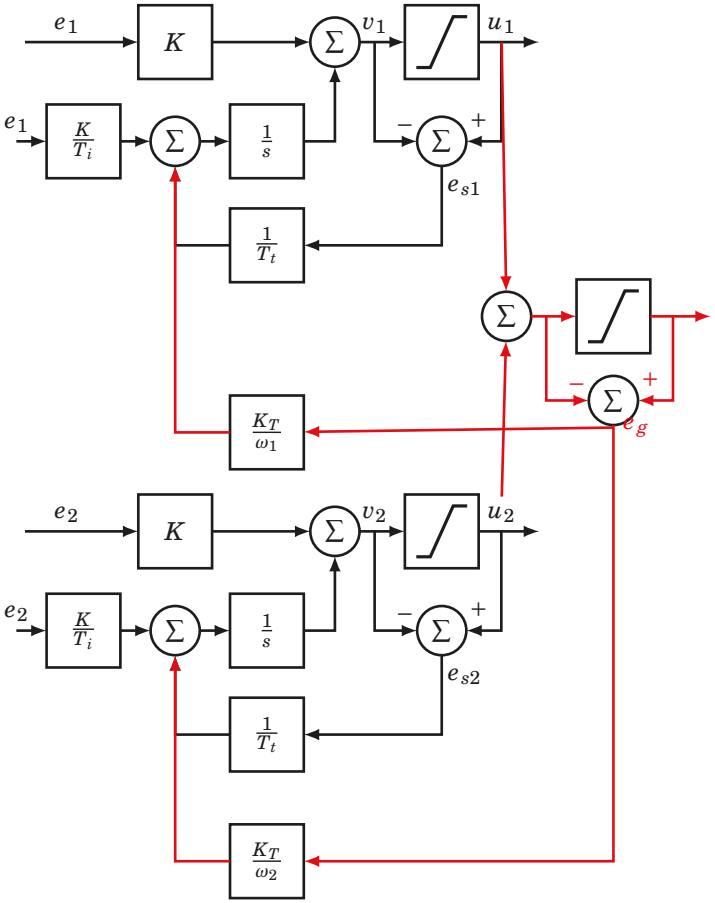


Figure 7. A tracking-based approach for handling global control signal constraints. The figure shows the case for two local controllers.

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 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

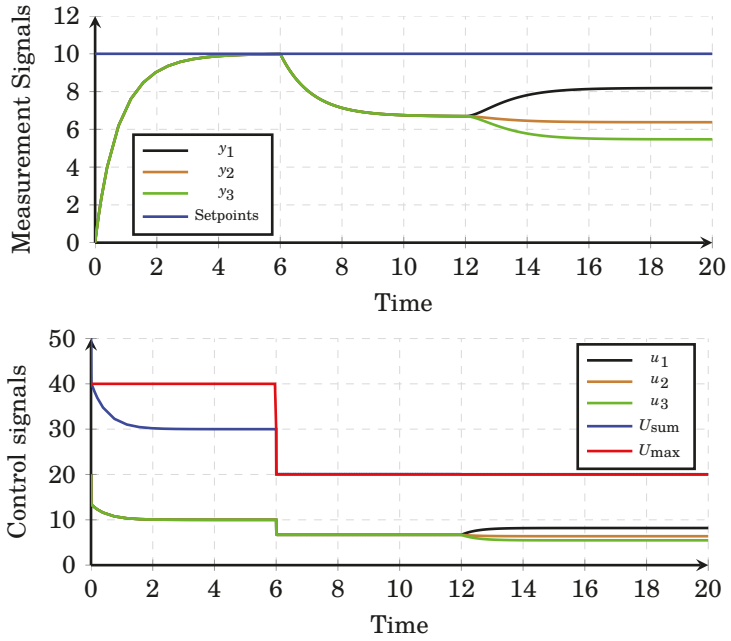


Figure 8. Three identical controllers which each has a control signal equal to 10. The upper plot shows the set-points (all equal) and the measurement signals whereas the lower plot shows the control signals, the sum of the control signals (in blue) and U_{max} in red. At time 6 the resource constraint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

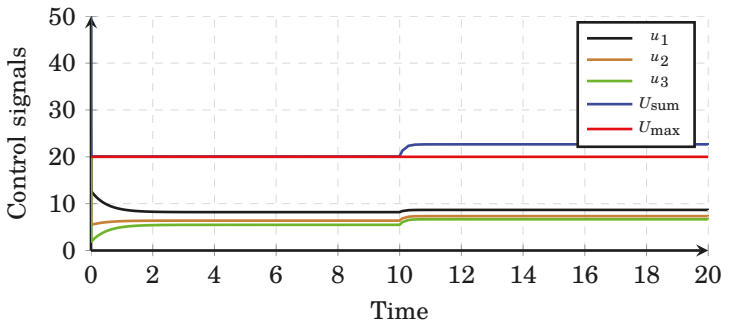


Figure 9. The same example as in Fig. 8 with only the control signal information shown. At time 10, K_T is changed from 100 to 1.

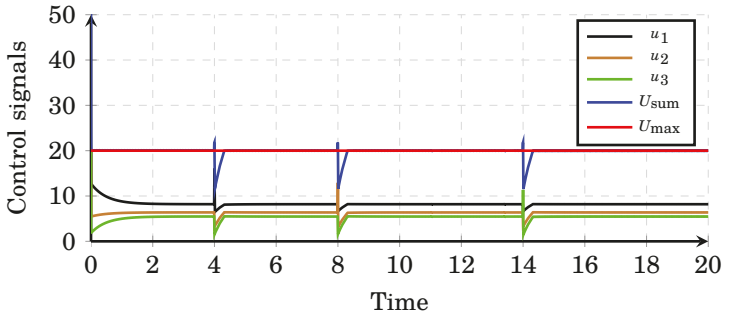


Figure 10. The same example as in Fig. 8 now with $K_T = 100$ all the time but with input pulse disturbances.

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.

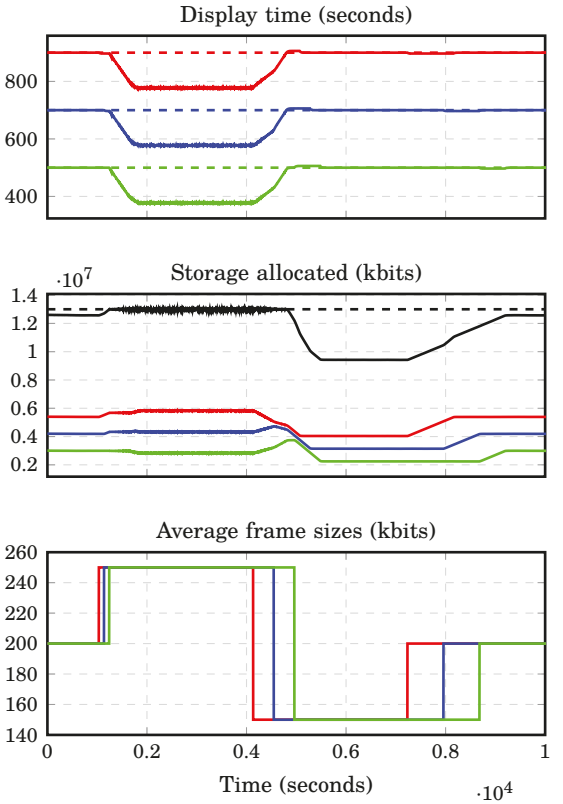


Figure 11. Stored video time, allocated storage, and average frame size. The weights of the control loops are identical. The red signals are for Camera 1, the blue for Camera 2 and the green for Camera 3. The sum of the control signals is shown in black and the constraint in dashed black.

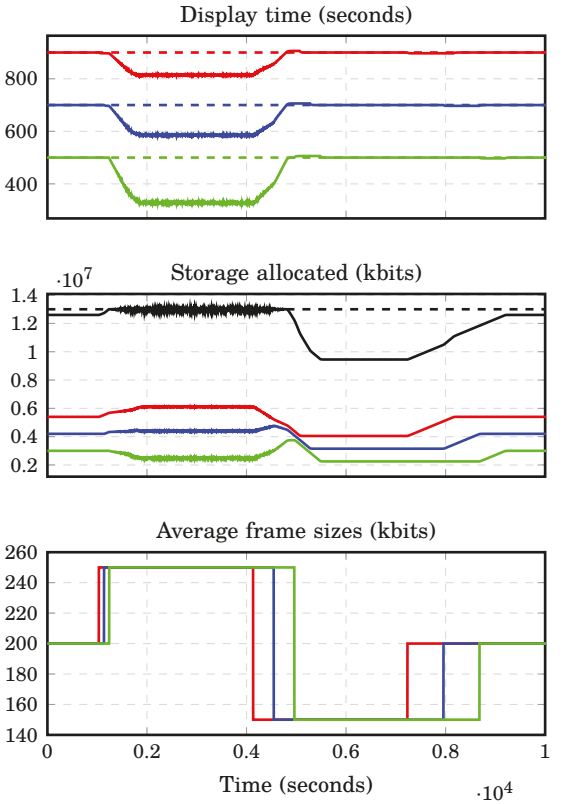


Figure 12. Stored video time, allocated storage, and average frame size with different weights ($\omega_1 = 100, \omega_2 = 133.3, \omega_3 = 200$). The red curves are for Camera 1, the blue for Camera 2 and the green for Camera 3. The sum of the control signals is shown in black and the constraint in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

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.

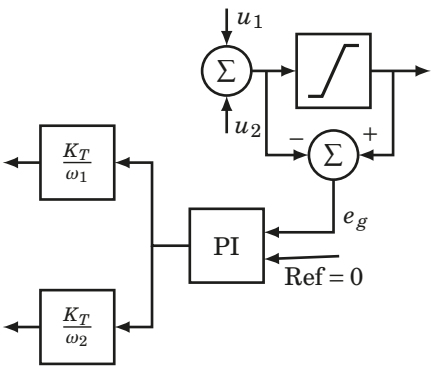


Figure 13. PI-based Tracking. The blocks should replace the right hand part of the block diagram in Fig. 7.

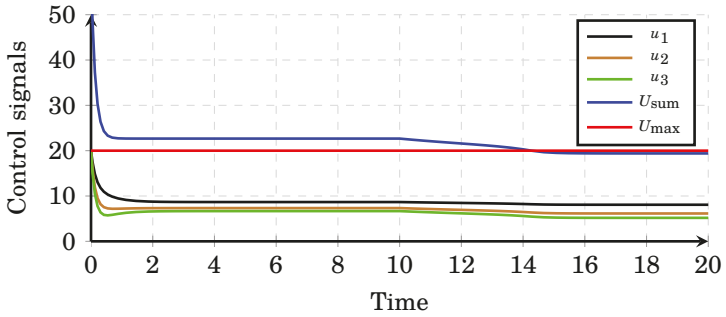


Figure 14. The same example as in Fig. 9 with only the control signal information shown. At time 10, PI-based global tracking is activated.

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

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

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

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

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

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

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

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

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

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

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

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

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

Wittenmark, B., K. Åström, and K.-E. Årzén (2003). *Computer Control: An Overview*. Tech. rep. Department of Automatic Control, Lund University.

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

Paper III

Dynamic Management of Multiple Resources in Camera Surveillance Systems

Alexandre Martins Karl-Erik Årzén

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.

1. Introduction

Computer and communication systems are interesting and challenging application areas for control, e.g., [Pothukuchi et al., 2020]. 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 [Martins et al., 2020] 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 [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 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

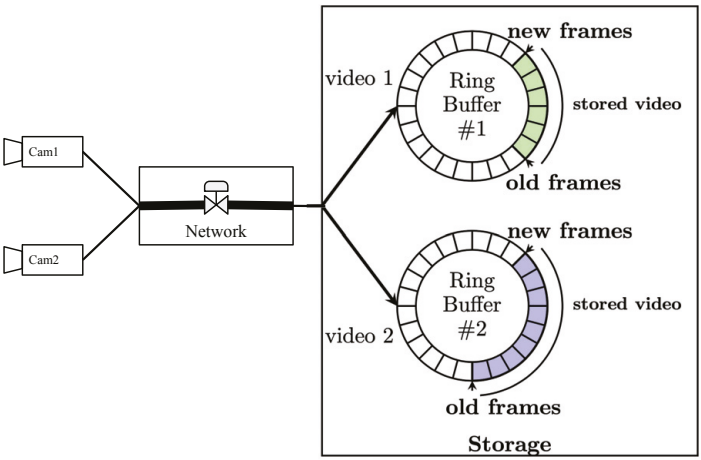


Figure 1. System overview.

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 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 [Martins et al., 2020] can be used to control two dependent shared resources.

1.1 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 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 [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 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.

1.2 Outline of the paper

Camera surveillance systems and how they have been modeled in this paper are discussed in Section 2. The control system architecture including the network bandwidth and storage controllers are presented in Section 3. Simulated evaluation results are given in Section 4. Finally, conclusions and suggestions for future work are discussed in Section 5.

2. 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 [Richardson, 2010]). 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 [Edpalm et al., 2018] and developed in detail in [Edpalm et al., 2018]. 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,

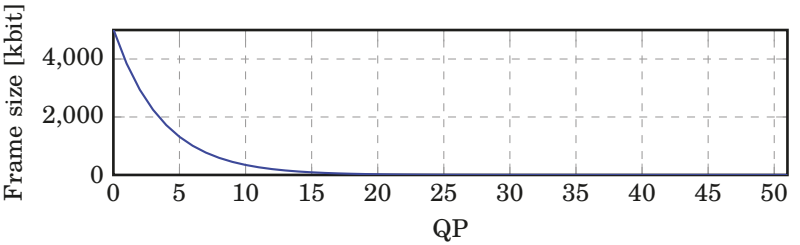


Figure 2. H.264 average video frame size for different QP (compression) values.

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 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.

2.1 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 [Harahap et al., 2016]).

The camera model consists of the encoder based on the model in [Edpalm et al.,

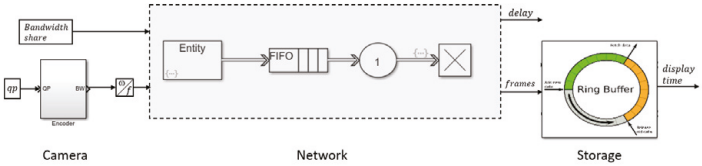


Figure 3. The overall structure of the model.

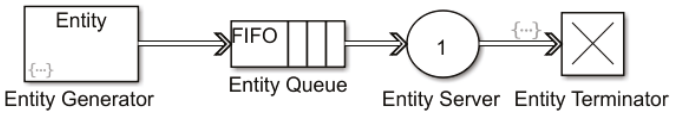


Figure 4. The SimEvent model of a network channel.

2018]. 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 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 [Martins et al., 2020]. 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.

3. Control System Architecture

The control system architecture is based on a combination of cascaded PI controllers, mid-ranging, and the tracking-based approach to handling shared resources proposed in [Martins et al., 2020]. The PI controllers are all discretized versions of the standard PI controller given by

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

3.1 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).***

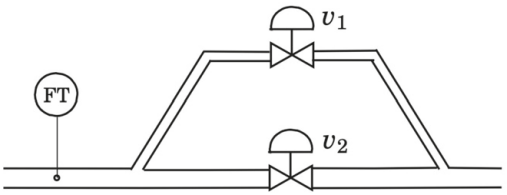


Figure 5. Mid-range application example.

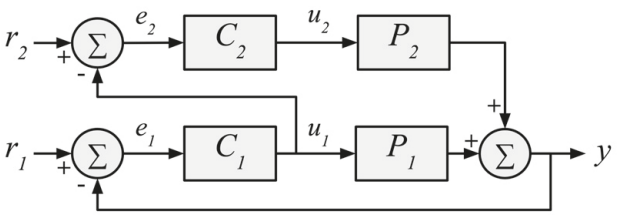


Figure 6. Mid-range controller structure. (this figure has been replaced due to errors, please see at the end of this article for details)

3.2 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 [Åström and Hägglund, 2006]. 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.

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 [Åström and Hägglund, 2006] pg 379. This has, however, not been considered necessary in our case.

3.3 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 [Martins et al., 2020]. 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 ([Åström and Murray, 2008]). It is assumed that the amount of shared resource that should 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

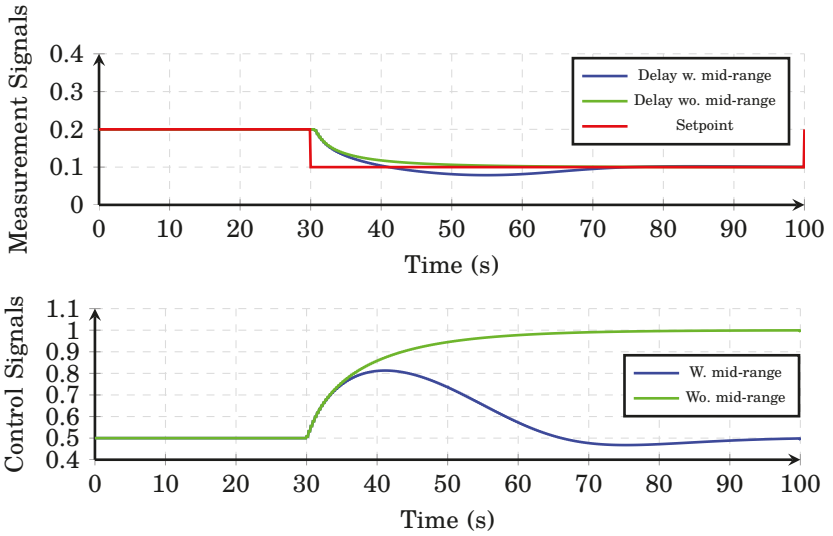


Figure 7. Response to a setpoint change in the delay without (wo.) and with (w.) mid-range control.

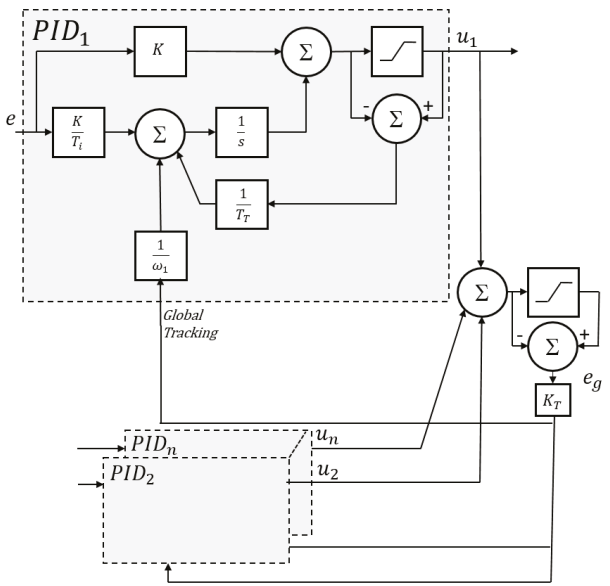


Figure 8. Tracking-based resource sharing.

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.***

3.4 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***.

3.5 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.

3.6 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.

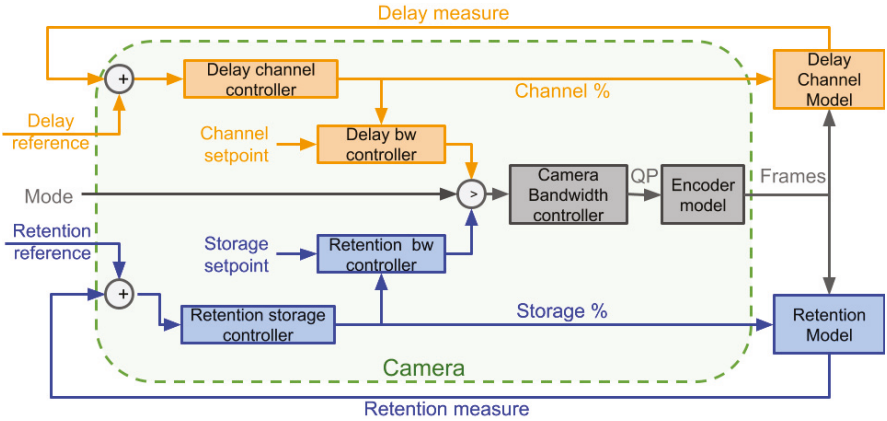


Figure 9. A block diagram of the overall controller structure.

4. 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

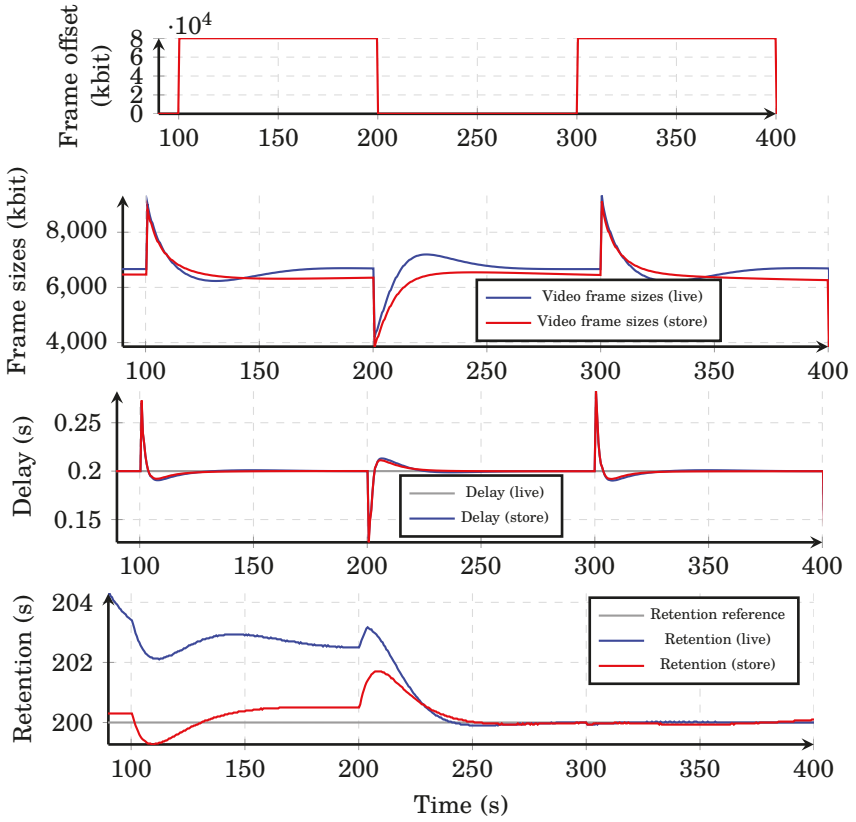


Figure 10. Scenario 1: Delay and retention for two cameras with the same parameters. One uses live mode and the other store mode.

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 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

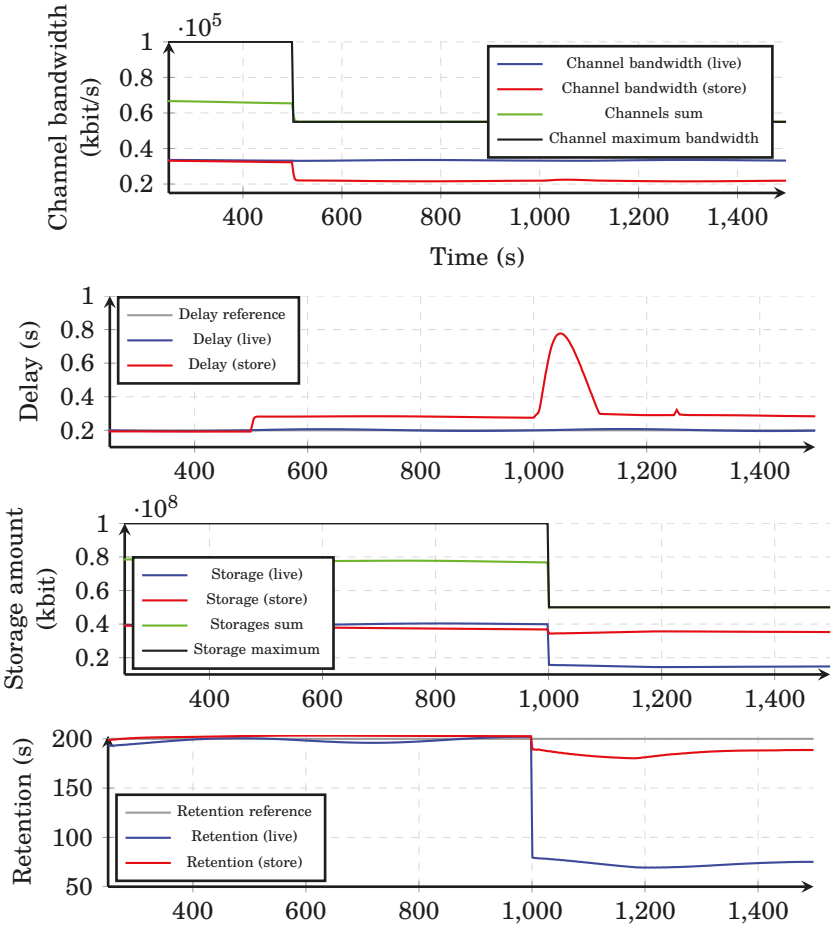


Figure 11. Scenario 2: Delay and retention for two cameras with the same parameters with live and store mode, with and without global limitations.

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.

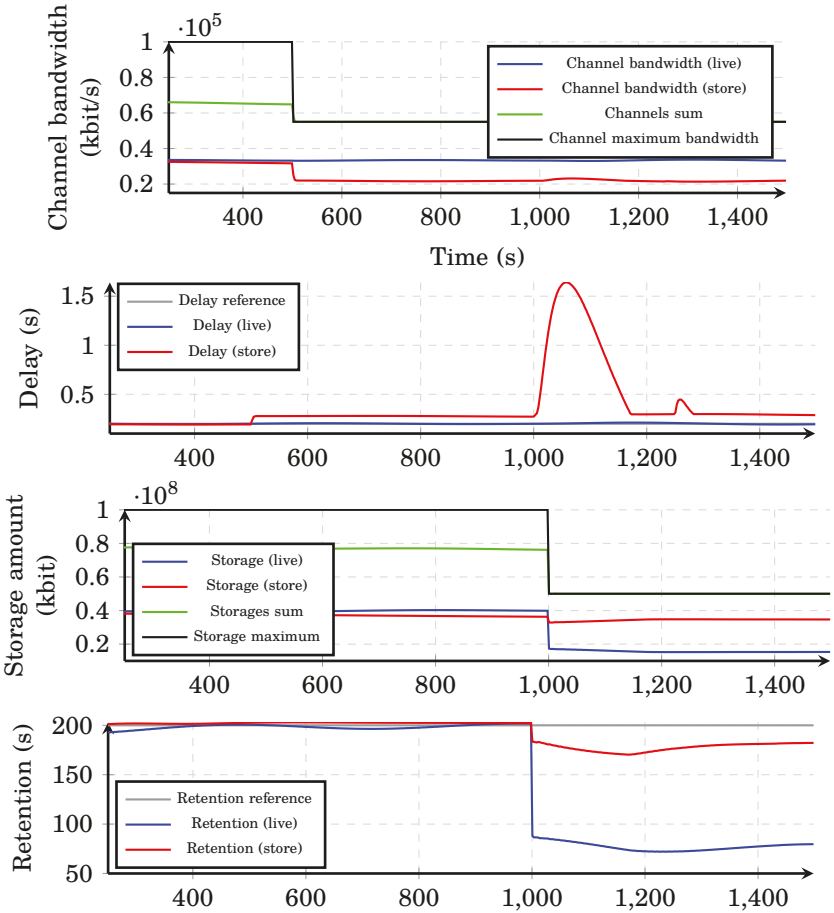


Figure 12. Scenario 3: Delay and retention for two cameras with the same parameters with live and store mode, with and without global limitations and different global resource priority.

The camera using live mode 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.

5. 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).

5.1 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.

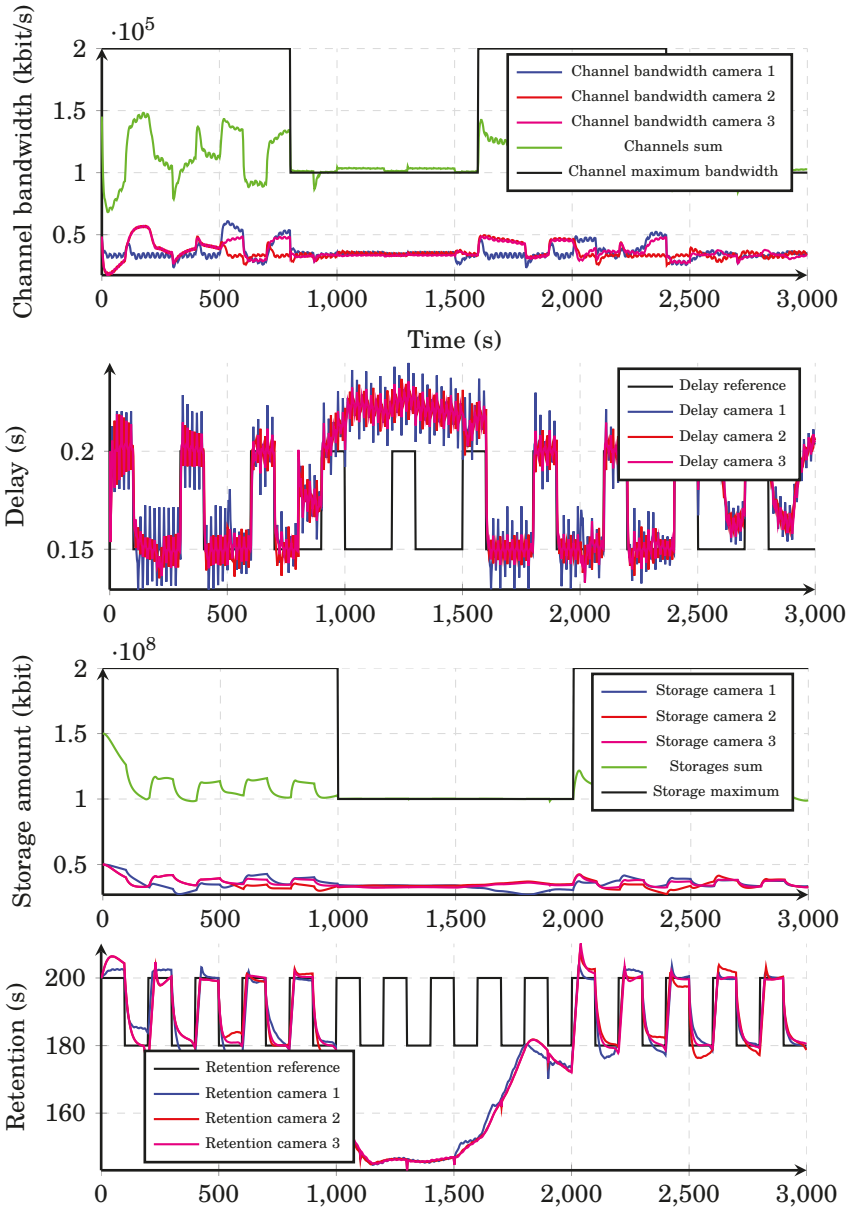


Figure 13. Scenario 4: Dynamic scenario - frame disturbance, delay, network bandwidth, retention, storage and modes changes, priorities are the same

References

Åström, K. J. and T. Hägglund (2006). *Advanced PID control*. ISA, Advanced PID control.

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

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

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

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

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

Edpalm, V., A. Martins, M. Maggio, and K.-E. Årzén (2018b). *H.264 Video Frame Size Estimation*. Technical Report. Department of Automatic Control, Lund University, Sweden.

Harahap, E., I. Sukarsih, G. Gunawan, M. Y. Fajar, D. Darmawan, and H. Nishi (2016). "A model-based simulator for content delivery network using simevents matlab-simulink". *INSIST* **1**:1.

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

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

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

Martins, A., M. Maggio, M. Lindberg, and K.-E. Årzén (2020). "Control-based resource management for storage of video streams". In: *IFAC World Congress, Berlin, Germany*.

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

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

Pothukuchi, R. P., S. Y. Pothukuchi, P. G. Voulgaris, and J. Torrellas (2020). "Control systems for computing systems: making computers efficient with modular, coordinated, and robust control". *IEEE Control Systems Magazine* **40**:2.

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

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

Change from original publication:

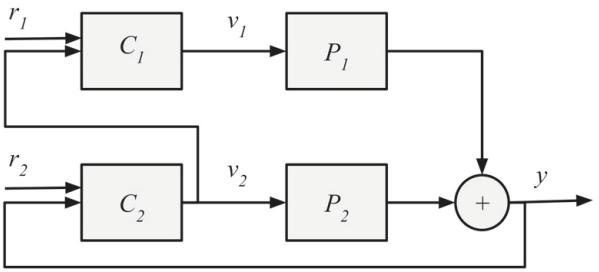


Fig.6 was wrong in the original publication in ACC 21. The original figure is shown here. The difference is that C_1 and C_2 have switched place in the block diagram. The same holds for P_1 and P_2.

Paper IV

Storage Allocation for Camera Sensor Networks using Feedback-Based Price Discrimination

Alexandre Martins Hung-Yu Wei Karl-Erik Årzén

Abstract

Camera sensor networks, mainly with surveillance cameras, are growing in size and complexity. Storage space is the prime resource in such systems but current surveillance setups are still very much centralized and limited in resources due to cost and security constraints. Allocating the correct amount of storage to each camera sensor considering their large difference in characteristics and video content is challenging. In this paper we propose a framework using feedback-based price discrimination of storage resources in order to guarantee a uniform quality level of the videos in camera sensor networks, regardless of the specific camera sensor parameters. We designed a lightweight solution using simple video quality metrics, cascade control and PI (Proportional and Integral) controllers to define the optimal price of resources per camera.

1. Introduction

The number and size of camera sensor systems used, *e.g.*, in different types of public spaces with surveillance cameras, are growing due to the Internet of Things (IoT) trend and they are currently one of the major storage and bandwidth consumers. With growing demands on high resolution, high frame rate and level of detail, the amount of storage needed to retain these videos is a growing problem. Surveillance installations are usually critical installations and are mostly running on dedicated infrastructures, storing video in trusted servers owned by systems administrators. Newer installations are usually large scale (commonly hundreds of cameras), heterogeneous and have large differences in resource requirement. [IPVM, 2021].

In this paper we propose a lightweight solution using the price discrimination principle from micro-economics, [Armstrong, 2008], to allocate storage resources while separating the resource providers (*i.e.*, the storage units) from the resource buyers (*i.e.*, the camera sensors). The buyers have private information on the amount of resources needed and act accordingly to maximize their utility (here the desire to minimize the compression of their own video stream). The utility represents the goal the buyers want to achieve. The storage units enforce the constraint on resource availability through the use of pricing.

The focus in this paper is H.264 video cameras, the dominating system on the market today. H.264 is a video compression standard based on block-oriented, motion-compensated coding [ITU-T, 2010]. A model of the bandwidth needed/generated by a H.264 surveillance camera was presented in our earlier work [Edpalm et al., 2018; Edpalm et al., 2018]. This model provides an estimate of the bandwidth needs for a H.264 video given current scene conditions and specific sensor parameters and allows to calculate the long term resource needs for the camera as long as it maintains the current parameters.

For a video surveillance system operator, the most important metric is the video quality. As such they want to have the best possible system-wide video quality given the current (mostly cost) constraints without knowledge of the prior or current characteristics of each camera sensor. The video compression level of H.264 videos, qp, determines the quality of each frame. The lower the qp value, the less compression is applied to the frame, the better the quality but the higher the frame size. The qp value and its variation over time have a direct impact on the perceived video quality (using mean opinion score testing) according to [Xue et al., 2010; Xue et al., 2013; Lin et al., 2012], *i.e.*, the lower and less varying the compression parameter qp is, the better the perceived quality will be. Our aim is thus to have all video cameras in the system to deliver videos with the same compression parameter values without having specific information about them.

The contributions of the paper are:

- A new flexible framework for facilitating resource allocation in medium- and large-size camera sensor networks.
- The use of cascade control to decide the price of resource per camera (price discrimination) so that the storage usage is maximized.
- A proposed utility measure for camera sensor networks based on the compression value (qp) and its deviation from a nominal value.

2. Related work

Price discrimination is a known profit optimization method in economics, [Armstrong, 2008], but it has been mostly used for revenue maximization. A study of different pricing schemes for maximizing revenue from selling cloud resources can be found in [Xu and Li, 2013]. [Li et al., 2009] studied the maximum revenue achievable by a monopolistic service provider under complete network information. Revenue maximization using price discrimination for communication network service providers was studied in [Shakkottai et al., 2008]. Price discrimination was used in order to distribute energy between sensors in [Edalat et al., 2009]. In [Tsakalozos et al., 2011] the same technique was used to optimally allocate virtual machines in a cloud service infrastructure. But, to the best of the authors knowledge, however, no prior work has used price discrimination in camera sensor networks for visual quality maximization.

Some centralized bandwidth allocation techniques optimizing the system's compression level have been proposed in [Seetanadi et al., 2018] and [Silvestre-Blanes et al., 2011]. Centralized task allocation for collaborative radar sensors based on resource availability and Quality of Service are proposed in [Yan et al., 2021] and [Giannecchini et al., 2004]. An alternative but related distributed approach to assign resources are auctions, thus second price auctions have been applied to video surveillance systems to optimize specific applications such as area overage [Chong Ding et al., 2012; Konda et al., 2016; Dieber et al., 2011], sensor placement [Elhamifar and Vidal, 2009; Ermis et al., 2010] and object tracking [Qureshi and Terzopoulos, 2009; Sankaranarayanan et al., 2008]. Auction theory has also been used to minimize content delivery delay and caching cost for large mobile networks involving multiple stakeholders as reported in [Li et al., 2016] or [Ghosh et al., 2004] or to allocate tasks between radar sensors as in [Ostwald et al., 2005].

3. Architecture, valuation & framework

We consider a simplified camera sensor network with one storage unit (Network Attached Storage, Cloud storage or other) and I IP video cameras, each having a camera sensor, indexed with i: $\{C_1, C_2, ...C_I\}$. An overview of the system with $I = 4$ is shown in Figure 1. Typically a video surveillance camera system is owned by a security department, which buys/rents storage from an IT department or cloud provider at a fixed rate. In our system, viewing quality is most important. The main system goal is to maximize the overall global video quality given the current system constraints: running cost and video storage size. The shared information between the devices and the system load should be as low as possible. We therefore use the video compression factor as a computation-free and simple way to measure the video quality. The direct correlation between the perceived video quality and the compression factor and its oscillation over time was studied in [Xue et al., 2013] and [Xue et al., 2010]. In H.264 videos, the compression factor is defined by the quantization parameter, $qp \in \{0, 1, \ldots, 51\}$ with 0 being lossless and 51 being the highest compression level [ITU-T, 2010]. The quantization controlled by the qp value is the only non-reversible step in the H.264 compression/decompression process impacting the visual quality.

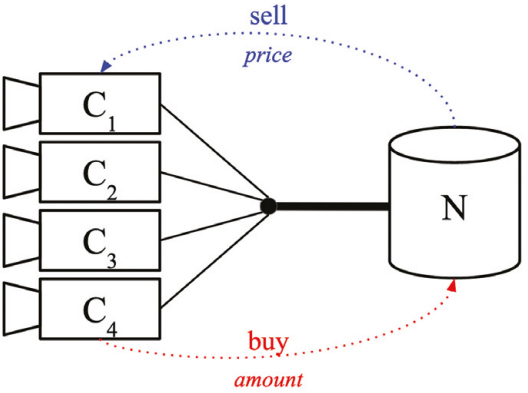


Figure 1. System with four cameras and one storage unit.

Every predefined period k, e.g., an hour, a day, or a week, the cameras need to buy storage from the seller to save the video they generate, using the money at their disposal. If the cameras run out of storage they need to wait until the next period to buy more. At the beginning of each period, k, the cameras obtain an amount of money, m, that they can use at their discretion to buy resources. The amount they receive depends on the cost of running the system. Each camera sensor has a virtual account holding the money it may use. Any remaining money can be saved for future periods. The amount of money available for camera C_i to buy storage at the beginning of each period k is: $m_i(k) = m_i(k-1) + m$. We do not enforce a limit to the amount of money a camera can retain if unused. How the money is distributed and enforced is not investigated in this paper.

It is assumed that all camera sensors in the network can communicate with the seller and they could, e.g., be part of the same virtual network. The total quantity of storage available by the storage provider is s and the storage space allocated to camera C_i is s_i. The corresponding expected quantities are annotated with a * superscript, *e.g.* the expected allocated storage s_i to camera C_i is denoted s_i^*. Only the storage unit has storage space, *i.e.*, the cameras are not storage providers.

3.1 Price and valuation of resources

Storage providers. The running price of each storage unit (Tbyte, Gbyte, etc.) is determined by the storage provider. The most common approach is to use marginal pricing, *i.e.*, the price is defined as the running cost plus a revenue margin. The storage provider will then charge $p_0 = p_0^{\min} + \epsilon$ where $p_0^{\min}$ is the running cost and ϵ the revenue margin. If $p_0 \leq p_0^{\min}$ the storage provider would sell at a loss. We can calculate $p_0^{\min}$ from the physical cost of hard disks, *e.g.*, a 8Tb hard disc costs around 400$, thus $p_0^{\min} = 0.05$ $ per Gb of storage. By adding a 20% margin, we would have $p_0 = 0,06$$.

In our approach, the seller will instead set different (or discriminate) prices per buyer based on the quality of the video stored by the buyer. We define the discriminate

price of camera i at time k as $p_i(k) \geq p_0^{\min}$

We denote with $R(k)$ the revenue of the seller at time k:

$$R(k) = \sum_{i=1}^{I} s_i(k) \cdot p_i(k) \tag{1}$$

where $p_i(k)$ is the price set by the seller and $s_i(k)$ is the amount of storage bought by camera i at time k.

The seller wants to maximize the camera's video quality given the current system constraints and to adjust the price to reflect the storage limitations without sacrificing the revenue.

The compression level, i.e., qp, of H.264 videos is part of the headers of the received videos. Hence, in each transaction period the storage provider has access to the qp of the received videos.

The discriminate prices are set with the help of PI controllers (one per camera) which compute the offset $\Delta p_i(k)$ to the running price p_0, *i.e.*, the discriminate price is $p_i(k) = p_0 + \Delta p_i(k)$. Proportional and Integral (PI) control is the most widely used control scheme in industry [Wittenmark et al., 2003]. The equation for a continuous-time PI controller is given by

$$u(t) = K\left(e(t) + \frac{1}{T_I}\int^t e(s)\,ds\right), \tag{2}$$

where $u(t)$ is the control signal, $e(t)$ the error between the desired value (or setpoint) of the measured signal and the actual value of the measured signal, and K and T_I are constant parameters. The storage provider has one controller per camera. It uses the current compression level, qp, of the video as the measured signal. The setpoint is determined by a single outer-loop probing controller which monitors the amount of storage allocated. The goal of the probing controller is to compute the desired compression level for all the cameras so that the storage usage is maximized. Probing control is a simple version of extremum-seeking control that is commonly used in process control, e.g., [Akesson and Hagander, 2000] and [Dochain et al., 2011]. The probing controller adjusts its output signal gradually until it reaches a good enough value, probing a new value at regular intervals to check if the new optimal value has changed.

Here the output of the probing controller is the desired system compression level which is used as the setpoint of the inner-loop PI controllers. The output of the PI controllers, *i.e.*, the control signal, is the discriminate price offset $\Delta p_i(k)$. In order to maximize the used storage the compression level should be as small as possible. Hence, the probing controller will decrease the desired compression level until the requested storage is at or above the maximum storage available. Then it will increase the desired compression level until the requested storage is within a safety margin and then keep it constant. It is kept constant until either (1) the requested storage is again at or above the maximum storage available, in which case it will start to increase the desired compression level again, or (2) a time-out event occurs, in which case it will again start to gradually decrease the desired compression level.

The cascade architecture with n cameras is shown in Figure 2 and the state machine of the probing controller is shown in Figure 3. The sampling period of the

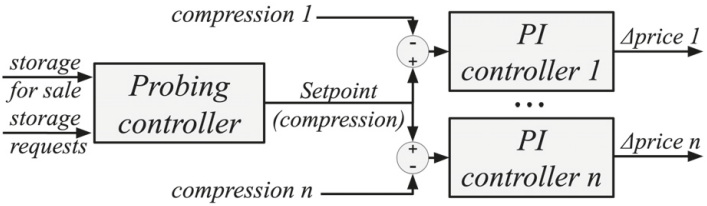


Figure 2. Cascade control structure, one probing controller decides the set-point of the price controllers.

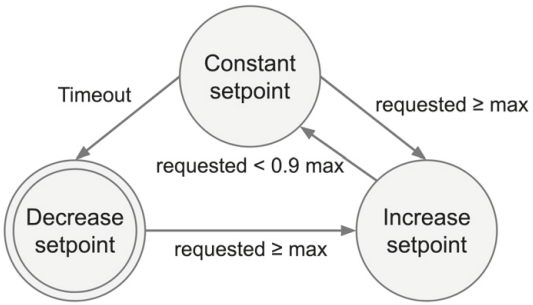


Figure 3. Probing controller state machine. The double border indicates that this is the initial state. The margin avoids rapid state changes close to the maximum storage amount. The Timeout event occurs when the Constant setpoint state has been active longer that a specified interval.

controllers is the transaction period and they are executed at the beginning of each period.

The effect of the feedback-based price discrimination is that the compression levels, qp, of the cameras will converge to the setpoint value of the PI controllers, *i.e.*, the value set by the outer probing controller.

If the total amount of storage requested by the cameras exceeds the total amount of available storage for sale, the storage provide will provide each camera C_i with an amount of storage s_i proportional to its demand compared to that of the other cameras.

Cameras. In order to decide how much storage a camera C_i wants to buy it needs to know how much storage it needs to store a video of a certain quality. An estimate of the storage needed for each qp is obtained using the frame size estimation model provided in [Edpalm et al., 2018]. This model is based on empirical values from multiple real surveillance videos. We denote the estimated storage at the period k for camera i, $s^*_{i,k}(qp)$, it provides for each qp the expected amount of storage necessary for a video with the current parameters (*e.g.*, motion in the scene, light level, amount of nature) and settings of the specific camera sensor (*e.g.*, frame rate, group of picture length). An example of $s^*_{i,k}(qp)$ is shown in Figure 4. The higher the qp is, the smaller

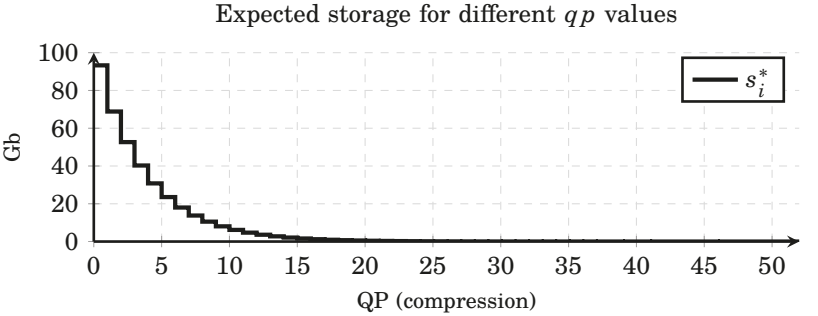


Figure 4. $s^*_{i,k}$ example.

the amount of storage needed and the lower the visual quality of the video.

At the beginning of each transaction period k, each camera C_i calculates $s^*_{i,k}(qp)$, i.e., an estimate of the storage need for each $qp \in \{0,1,\ldots,51\}$ given the actual scene and camera sensor parameters which are assumed to be measured or estimated by the camera. The $s^*_{i,k}(qp)$ functions differ from camera to camera and over time because each camera sensor which equips camera C_i has different settings and overlooks a different non-constant scene.

Camera C_i uses the actual curve to decide how much storage it should buy with its available money $m_i(k)$, see Section 3.1. We do not impose any limitation on the saved funds of cameras and unused money could be saved indefinitely.

Camera utility. The more storage the camera has, the lower qp it can use to compress its video and therefore the better the video quality [Xue et al., 2010] will be. Oscillations between qp values have a large impact on the visual quality of the video because of the visible jumps in visual quality [Xue et al., 2013].

The valuation function θ_i of buyer C_i designed to embody the system objective, *i.e.*, to retain videos of the highest possible quality in the system, where quality is measured by the video compression level, qp_i, and how much it varies. It is defined by the ellipse equation

$$\theta_i(qp_i, m_i) = m_i \cdot \sqrt{1 - \left(\frac{qp_i + \sigma_n(qp_i)}{2 \cdot 51}\right)^2}, \tag{3}$$

where m_i is the money available for the camera C_i, qp_i the compression value corresponding to the received amount s_i, and $\sigma_n(x)$ is the standard deviation of x over the n last periods.

The equation of an ellipse has an interesting characteristic around its vertexes. The derivative of the ellipse is low when approaching the co-vertex (low qp and low $\sigma(qp)$), while it is high when close to the vertex (high qp and high $\sigma(qp)$). It is valued more (high derivative) to move away from the high qp and high $\sigma(qp)$ values (vertex) than it is to get closer to the low qp and low $\sigma(qp)$ values (co-vertex).

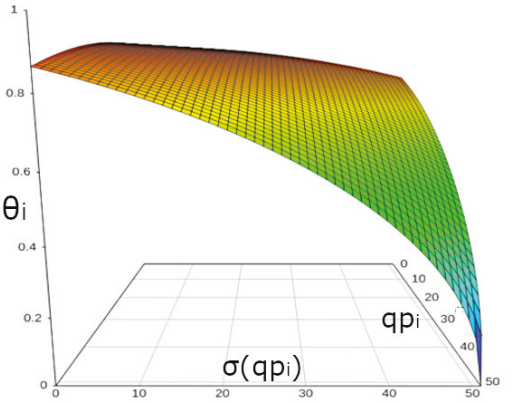


Figure 5. An illustration of how the valuation of resources, θ_i, depends on the compression and its variation.

The utility u_i of buyer C_i is then given by

$$u_i(qp_i, m_i, p_i) = \theta_i(qp_i, m_i) - p_i \tag{4}$$

where p_i is the price paid to obtain the amount of storage s_i. The smaller the compression level and the variation of the compression level the higher the camera utility will be. An example of the utility is shown in Fig 5. We use the last 10 qp values from previous transactions to calculate the standard deviation. The longer this history, the longer a deviation in qp will affect the utility.

At the beginning of each transaction period k, after calculating the expected storage amount of storage $s^*_{i,k}(qp)$ (see Section 3.1), the camera calculates the expected utility u^*_i assuming that it gets the associated storage $s^*_{i,k}$ given the available money $m_i(k)$ and the announced unit price $p_i(k)$.

Different cameras can have different strategies for buying storage. Here we consider two possible strategies:

1. At each period k, the camera buys the amount of storage $s^*_{i,k}$ which maximizes the expected utility $u^*_i(k)$ given the money $m_i(k)$ available.
2. The camera keeps all the money until an event occurs. When the event occurs it acts according to Strategy 1 (above).

3.2 Transaction steps

We use a transaction mechanism inspired by the closed bid transaction mechanism used in auctions [Reck, 1997]. A transaction is defined by the step described below. At the beginning of each transaction period k:

1. Camera C_i gets an amount of money, m, for the new period k. The money available to the camera is $m_i(k) = m_i(k-1) + m$.

2. The storage provider, n, announces the total amount of storage for sale, $s(k)$, and the unit price of camera C_i: $p_i(k)$.
3. Camera C_i decides how much it buys based on the expected storage usage $s^*_{i,k}(qp)$, $p_i(k)$ and $m_i(k)$. It sends to n the amount from $s^*_{i,k}(qp)$ maximizing its expected utility $u^*_i(k)$ (see Section 3.1).
4. Storage provider n decides the storage allocation and sends to C_i the amount of storage it is allowed to buy, s_i.
5. Camera C_i pays the storage provider n the price $s_i \cdot p_i$, deduces this amount from the money it has, *i.e.,* $m_i(k) = m_i(k) - s_i \cdot p_i$, and starts streaming video data up to the provided amount s_i allocated.
6. Storage n extracts the compression level from the received videos, $qp_i(k)$, and uses it to decide the price for the next transaction $p_i(k+1)$ using the PI controllers. It also calculates the total amount of storage allocated, $\sum_i s_i(k)$, and uses it to adjust the desired compression level using the probing controller so that the storage usage is maximized, see 3.1.

4. Results

To validate the price discrimination approach we run multiple simulations using a python framework with independent players (seller and buyers) communicating via queues as well as simulations using the CORE real time network emulator [Ahrenholz et al., 2008]. Each simulation uses random unit prices, p_0, and random camera parameters (resolution and motion level). The storage needs of the cameras are determined using the model described in [Edpalm et al., 2018]. The cameras will have a computation horizon of 10 periods to calculate their utility.

In the simulations we compare three different cases:

1. The storage uses marginal pricing, *i.e.*, it defines the running cost and adds a margin to it (see Section 3.1).
2. The storage uses the price discrimination scheme described in Section 3 with a fixed system quality setpoint, i.e., without any probing controller.
3. The storage uses the price discrimination scheme described in Section 3 with the probing controller selecting the system quality setpoint.

The camera utility is given by Equation (4). We also define a seller utility $U(k)$ to visualise how efficiently the proposed approach reduces the standard deviation of the sum of the compression levels. It is given by

$$U(k) = \frac{1}{\sigma_n\left(\sum_i^I qp_i(k)\right)} \tag{5}$$

The simulations use four cameras in Fig 7 and Figure 9 and ten cameras in Figure 8. The storage price p_0 is set randomly at simulation start. In the simulations in Figure 7 and Figure 9, C_1 is a 4K camera and as such requires the largest storage amount, C_2 and C_3 are 1080p cameras with different scene characteristics (C_2's video is more noisy

and has more motion) and C_4 is a 720p camera. In Figure 8, camera parameters are randomly selected at simulation start. During the simulations the camera parameters are constant (resolution and motion levels are fixed) but uniform noise of amplitude up to 25% of the frame sizes was added to reflect a real scenario where noise from the sensor and small changes in the scene would create changing frame sizes. All cameras receive the same amount of money m at each period k. Figure 7 shows the simulation results of case 1 and case 2. The left column contains the results using marginal pricing scenario (case 1) and the right column contain the results using price discrimination with PI control but without probing controller, i.e., with a fixed setpoint (case 2).

The uppermost plots contain the qp values of the cameras (remember that a higher qp means a lower video quality), the ones below are the prices set by the storage provider (gray for the base price, colored for discriminate prices). The third row is the camera utility and the final two rows show the seller utility (defined in 1) and the revenue from selling storage to the cameras (see 3.1). Note that in the utility plots the maximum utility has been limited to 5 for easier plotting and calculations (as the utility of an infinitesimal standard deviation tends to infinity).

In the right column of Figure 7, we can see that the PI controllers change the unit prices p_i to allow the video compression qp_i to converge towards a common qp value (first row). The seller utility U (fourth row) in the price discrimination case (case 2, right column) is clearly higher than in the fixed price case (case 1, left column) while the seller revenues R are very close to each other (fifth row), *i.e.*, the price discrimination policy allows the total system to run in a better state than using marginal price policy. Because of the seller utility definition, the utility value will tend to infinity when the standard deviation of the camera qp is zero, leading to the jumps we can see in Figure 7.

In the simulations in Figure 9 price discrimination with probing controller (case 3) is used in both columns. The camera parameters (apart for the video resolution) are selected randomly at simulation start. The results in the left and right column are from two different runs. In the left column, cameras are buying storage at each period k, the simulation being run for 400 periods. In the right column the simulation is done with 4 cameras over 100 periods, cameras C_3 and C_4 are buying at each time period but C_1 and C_2 only buy every 5 and 12 periods respectively. The simulations were done using the same code and models as the python framework but the seller and buyers were running in virtual machines communicating through sockets in the CORE real time network emulator [Ahrenholz et al., 2008]. A screenshot of the used system can be seen in Figure 6. With the CORE simulator we can simulate multiple machines communicating over different network architectures and simulate different network conditions. We used the CORE simulator to ensure communication was not sequential and reflected a real-world setup without having to deploy such a setup. The focus of the python framework is to simulate the system sequentially with a focus on the global system behavior.

In the right part of Figure 9 the seller revenues oscillate because of the less frequent storage buy from cameras C_1 and C_2, but the average revenue is comparable and the different strategies does not prevent the system from converging to a common compression level. In the left part of Figure 9, we can observe the effect of the

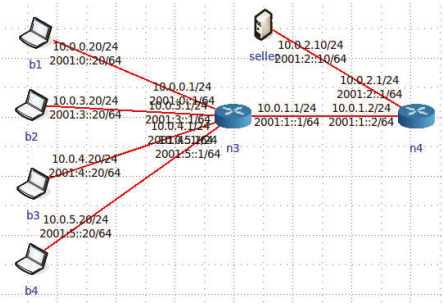


Figure 6. CORE network emulator setup.

probing controller adjusting gradually the setpoint compression level down in order to maximize the storage usage, leading to a stable system value with $qp = 20$. We can also see that the prices set follows the same trend in order to converge to the system setpoint set by the probing controller but also each discriminate prices diverge in order for the compression of each camera to converge to a common value thanks to the price discrimination PI controllers.

Figure 8 shows simulations with 10 cameras and same selected parameters (randomly chosen once for both simulations) and we visualize the most interesting 250 periods. In this figure, the left side has only the price discrimination PI controllers (case 2) running with a manually fixed global quality setpoint of $qp = 25$ (which is the optimal setpoint for this specific system). The right side of the figure shows the same parameters with the setpoint controller (case 3) converging autonomously to the optimal value of $qp = 25$. In the left column (case 2) we can see that all cameras converge slowly to the defined setpoint while on the right (case 3) they converge faster and autonomously to the desired quality levels.

Finally, to test the robustness of the proposed approach, we run 100 simulations of 500 transactions period k each with price discrimination (case 3) and 100 others with marginal pricing (case 1) where all cameras were buying at each period k. We also run simulations with random uniform distributed video parameters changing from frame to frame. In reality this is highly unlikely to happen, but demonstrate a hypothetical worst case scenario. This is denoted with "+rnd" in Table 1.

A global summary of the simulation runs can be found in Table 1. The seller utilities U are on average (mean) higher in the discriminate pricing (1.68 and 0.56) than for the marginal pricing (0.33 and 0.27). Note that the utility difference between the random and non-random case in Table. 1 comes from the utility being based on $\sigma_n(qp)$. With randomly changing video parameters, the optimal qp value will rarely stay equal to the system setpoint qp value, thus leading to a lower utility value for the storage than in a more stable environment. The revenue R remains very close at 19.8 and 20 for the discriminate pricing versus 20 for the margin pricing indicating no noticeable loss in revenue for the seller. The buyer utilities u_i have higher values with price discrimination than with the marginal pricing because of the visual quality uniformity enforced by the storage provider.

	Discriminate	Discriminate+rnd	Margin	Margin+rnd
Seller utilities mean	1.68	0.56	0.33	0.27
Seller revenue mean	19.8	20	20	20
Buyer 1 utility mean	0.87	0.82	0.81	0.78
Buyer 2 utility mean	0.86	0.82	0.8	0.75
Buyer 3 utility mean	0.86	0.82	0.83	0.79
Buyer 4 utility mean	0.86	0.83	0.82	0.8

Table 1. Multiple simulations results.

5. Conclusions & Future Work

In this work we proposed a method based on price discrimination of storage costs for system-level optimization of video quality, which to the best of the authors knowledge is a novel approach to solve video storage allocation. The approach is lightweight and requires limited system knowledge and computation requirements. The results in terms of system-wide video quality are encouraging and do not lead to significant revenue loss for the storage sellers but improves the overall system video quality. The simplified approach also lays the ground for future development of game theory approaches using the same transaction framework.

A logical extension of this paper would be to handle multiple storage providers and develop more complex utility functions for both cameras and storage sellers which would take into account different constraints such as network latency and bandwidth. We could also use a different convergence method which would optimize the storage usage more by allowing the compression levels to slightly deviate for some cameras. Instead of having the probing controller increasing or decreasing the compression level for all the cameras it could increase/decrease the compression level of the cameras one at a time, ensuring that at all times the cameras have setpoints that maximally differ with one compression level value. This would increase the storage utilization.

A limitation of this work is that we expect the storage provider to be able to access the received videos in order to get access to the qp values in order to decide on a discriminate price. If the video is stored in an encrypted format this technique could not be used. Video quality is here considered as correlated to the video compression. An alternative approach would be to use an application specific metric or a recognized quality metric such as the structural similarity index measure (SSIM), peak signal-to-noise ratio (PSNR) or other metrics enumerated in [Yang, 2007], but at the expense of additional computation costs.

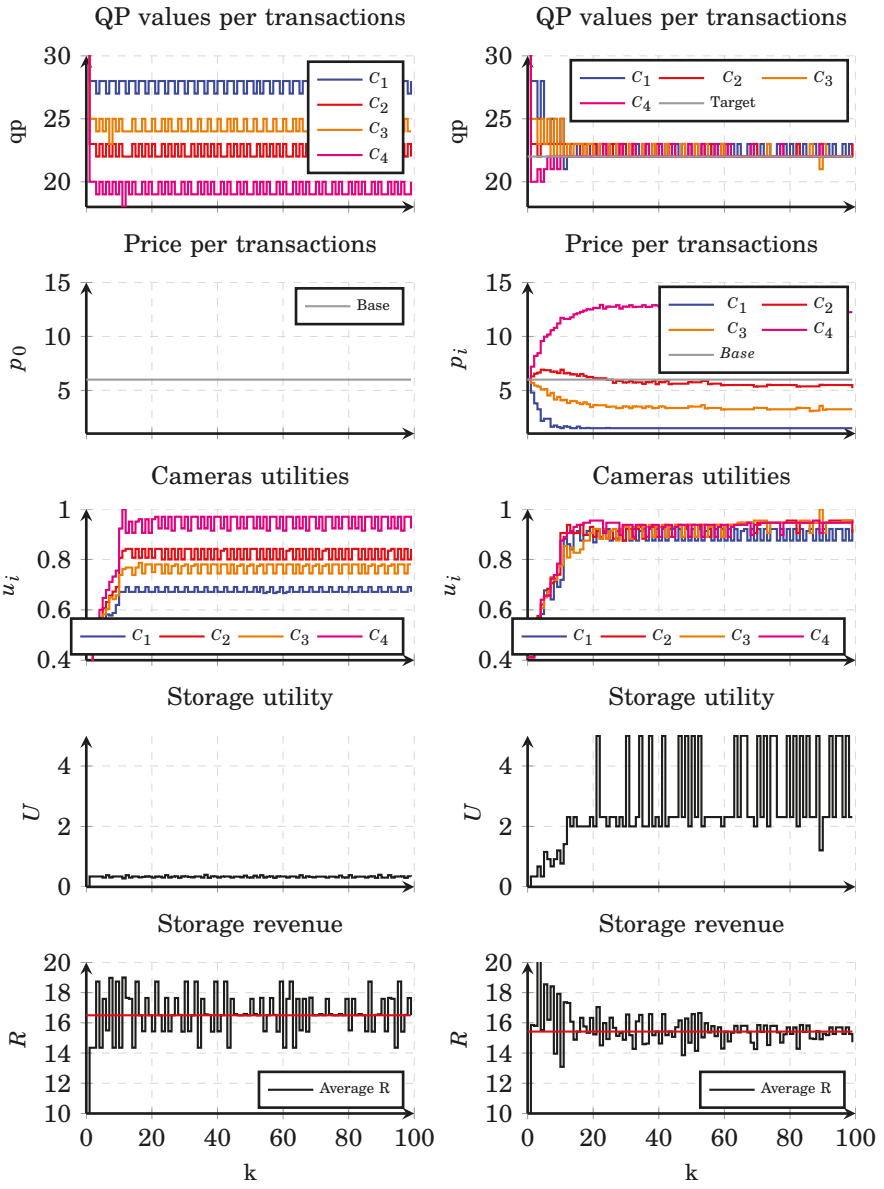


Figure 7. Fixed price (left) and discriminate price (right) with fixed system setpoint (no probing controller) for 4 cameras.

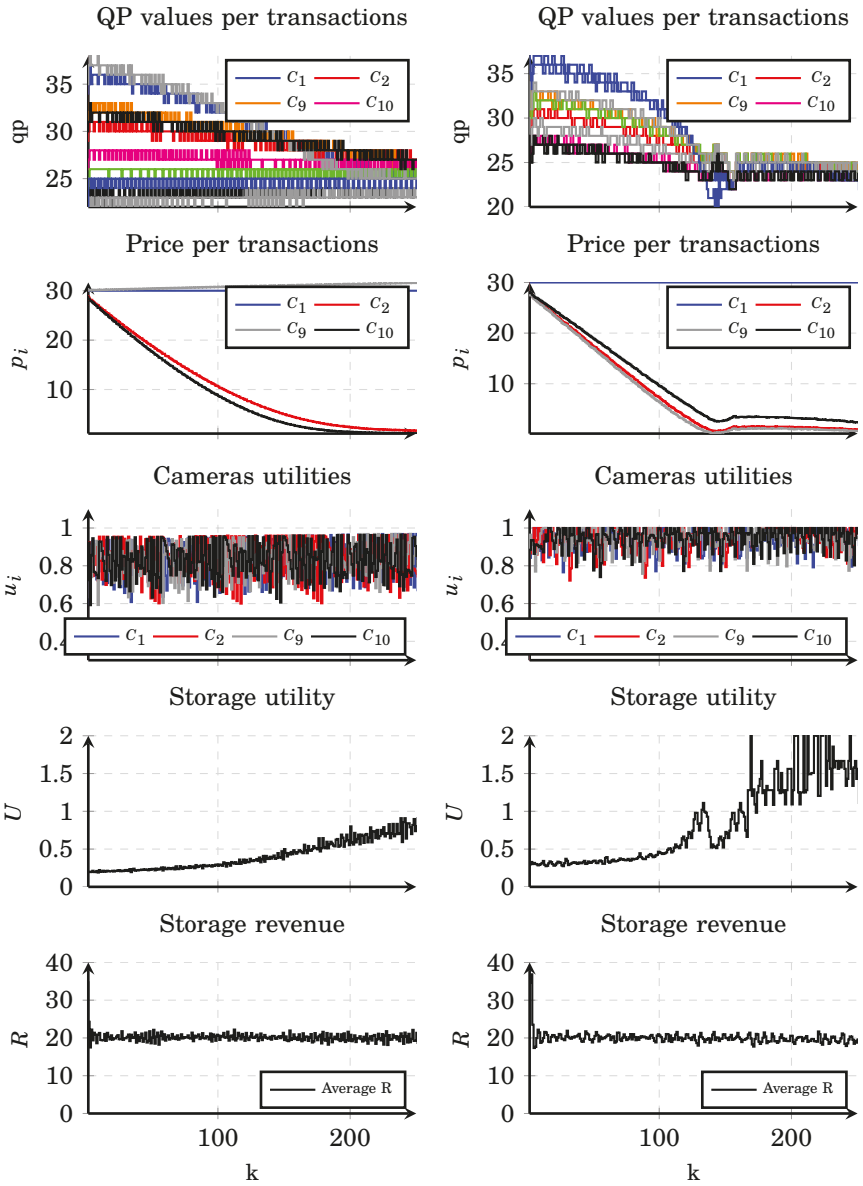


Figure 8. Discriminate price with fixed setpoint (left) and with probing controller (right) for 10 cameras.

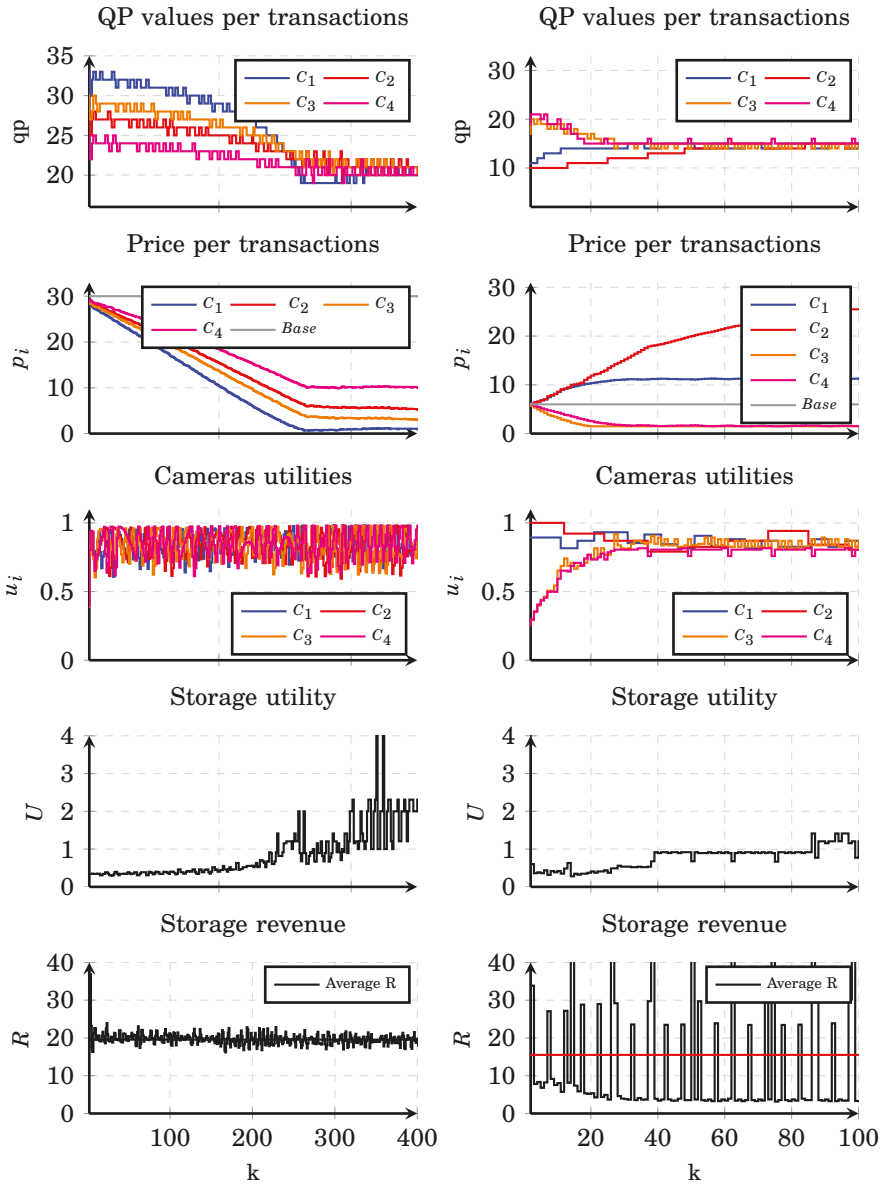


Figure 9. Discriminate price with setpoint probing controller for continuous buyers only (left) or continuous and event buyers (right) for 4 cameras.

References

Ahrenholz, J., C. Danilov, T. R. Henderson, and J. H. Kim (2008). "Core: a real-time network emulator". In: *MILCOM 2008-2008 IEEE Military Communications Conference*. IEEE, pp. 1–7.

Akesson, M. and P. Hagander (2000). "A simplified probing controller for glucose feeding in escherichia coli cultivations". In: *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*. Vol. 5, 4520–4525 vol.5.

Armstrong, M. (2008). *Price discrimination*. MIT Press.

Chong Ding, Bi Song, A. Morye, J. A. Farrell, A. K. Roy-Chowdhury, C. Ding, B. Song, A. Morye, J. A. Farrell, and A. K. Roy-Chowdhury (2012). "Collaborative Sensing in a Distributed PTZ Camera Network". *IEEE Transactions on Image Processing* **21**, pp. 3282–3295.

Dieber, B., C. Micheloni, and B. Rinner (2011). "Resource-aware coverage and task assignment in visual sensor networks". *IEEE Transactions on Circuits and Systems for Video Technology* **21**:10, pp. 1424–1437.

Dochain, D., M. Perrier, and M. Guay (2011). "Extremum seeking control and its application to process and reaction systems: a survey". *Mathematics and Computers in Simulation* **82**:3. 6th Vienna International Conference on Mathematical Modelling, pp. 369–380. ISSN: 0378-4754.

Edalat, N., W. Xiao, C.-K. Tham, E. Keikha, and L.-L. Ong (2009). "A price-based adaptive task allocation for wireless sensor network". In: *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, pp. 888–893.

Edpalm, V., A. Martins, K.-E. Årzén, and M. Maggio (2018a). *Camera networks dimensioning and scheduling with quasi worst-case transmission time*. Vol. 106. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing.

Edpalm, V., A. Martins, M. Maggio, and K.-E. Årzén (2018b). *H.264 Video Frame Size Estimation*. Technical Reports TFRT-7654. Department of Automatic Control, Lund Institute of Technology, Lund University.

Elhamifar, E. and R. Vidal (2009). "Distributed calibration of camera sensor networks". *2009 3rd ACM/IEEE Int. Conference on Distributed Smart Cameras, ICDSC 2009*.

Ermis, E. B., P. Clarot, P.-M. M. Jodoin, and V. Saligrama (2010). "Activity based matching in distributed camera networks". *IEEE Transactions on Image Processing* **19**, pp. 2595–2613.

Ghosh, P., N. Roy, S. K. Das, and K. Basu (2004). "A game theory based pricing strategy for job allocation in mobile grids". In: *18th Int. Parallel and Distributed Processing Symp., 2004. Proceedings.* Pp. 82–.

Giannecchini, S., M. Caccamo, and C.-S. Shih (2004). "Collaborative resource allocation in wireless sensor networks". In: *Proceedings. 16th Euromicro Conference on Real-Time Systems, 2004. ECRTS 2004.* Pp. 35–44.

IPVM (2021). *Top 5 problems in video surveillance storage*. URL: https://ipvm.com/reports/problems-video-surveillance-storage (visited on 2021-08-18).

ITU-T (2010). *H.264 standard documentation*. URL: https://www.itu.int/rec/T-REC-H.264 (visited on 2019-06-13).

Konda, K. R., N. Conci, and F. De Natale (2016). "Global coverage maximization in PTZ camera networks based on visual quality assessment". *IEEE Sensors Journal*.

Li, J., J. Sun, Y. Qian, F. Shu, M. Xiao, and W. Xiang (2016). "A commercial video-caching system for small-cell cellular networks using game theory". *IEEE Access* **4**, pp. 7519–7531.

Li, S., J. Huang, and S.-Y. R. Li (2009). "Revenue maximization for communication networks with usage-based pricing". In: *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. IEEE, pp. 1–6.

Lin, X., H. Ma, L. Luo, and Y. Chen (2012). "No-reference video quality assessment in the compressed domain". *IEEE Transactions on Consumer Electronics* **58**:2, pp. 505–512.

Ostwald, J., V. Lesser, and S. Abdallah (2005). "Combinatorial auctions for resource allocation in a distributed sensor network". In: *26th IEEE International Real-Time Systems Symposium (RTSS'05)*, 9 pp.–274.

Qureshi, F. Z. and D. Terzopoulos (2009). "Planning ahead for PTZ camera assignment and handoff". In: *2009 Third ACM/IEEE Int. Conference on Distributed Smart Cameras (ICDSC)*. IEEE, pp. 1–8.

Reck, M. (1997). "Trading-process characteristics of electronic auctions". *Electronic Markets* **7**:4, pp. 17–23.

Sankaranarayanan, A., A. Veeraraghavan, and R. Chellappa (2008). "Object Detection, Tracking and Recognition for Multiple Smart Cameras". *Proceedings of the IEEE* **96**, pp. 1606–1624.

Seetanadi, G. N., L. Oliveira, L. Almeida, K.-E. Arzén, and M. Maggio (2018). "Game-theoretic network bandwidth distribution for self-adaptive cameras". *SIGBED Rev.* **15**:3, pp. 31–36.

Shakkottai, S., R. Srikant, A. Ozdaglar, and D. Acemoglu (2008). "The price of simplicity". *IEEE Journal on Selected Areas in Communications* **26**:7, pp. 1269–1276.

Silvestre-Blanes, J., L. Almeida, R. Marau, and P. Pedreiras (2011). "Online qos management for multimedia real-time transmission in industrial networks". *IEEE Transactions on Industrial Electronics* **58**:3, pp. 1061–1071.

Tsakalozos, K., H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Paparas, and A. Delis (2011). "Flexible use of cloud resources through profit maximization and price discrimination". In: *2011 IEEE 27th International Conference on Data Engineering*. IEEE, pp. 75–86.

Wittenmark, B., K. Åström, and K.-E. Årzén (2003). *Computer Control: An Overview*. Tech. rep. Department of Automatic Control, Lund University.

Xu, H. and B. Li (2013). "A study of pricing for cloud resources". *ACM SIGMETRICS Performance Evaluation Review* **40**:4, pp. 3–12.

Xue, Y., Y.-F. Ou, Z. Ma, and Y. Wang (2010). "Perceptual video quality assessment on a mobile platform considering both spatial resolution and quantization artifacts". In: *2010 18th International Packet Video Workshop*. IEEE, pp. 201–208.

Xue, Y., Y. Song, Y.-F. Ou, and Y. Wang (2013). "Video adaptation considering the impact of temporal variation on quantization stepsize and frame rate on perceptual quality". In: *International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, pp. 70–74.

Yan, J., J. Dai, W. Pu, S. Zhou, H. Liu, and Z. Bao (2021). "Quality of service constrained-resource allocation scheme for multiple target tracking in radar sensor network". *IEEE Systems Journal*, pp. 771–779.

Yang, K.-C. (2007). *Perceptual quality assessment for compressed video*. PhD thesis. UC San Diego.

Paper V

Vickrey-Clarke-Groves Auction-Based Storage Allocation for Distributed Camera Systems

Alexandre Martins Hung-Yu Wei Karl-Erik Årzén

Abstract

Video surveillance systems are critical infrastructures and that growing in size and complexity. Storage space is the prime resource in such systems but current surveillance setups are centralized and limited in resources due to security and cost constraints. Allocating the correct amount of storage to each camera considering their large differences in characteristics and video content is challenging. In this paper we propose a game theoretic approach to storage allocation for video surveillance camera systems based on the Vickrey-Clarke-Groves (VCG) auction mechanism.

1. Introduction

The number and size of camera systems used, *e.g.*, in different types of public spaces, are growing due to the Internet of Things (IoT) trend and they are currently one of the major storage and bandwidth consumers. With growing demands on high resolution, high frame rate and level of detail, the amount of storage needed to retain these videos is a growing problem. Surveillance installations are usually critical installations and are mostly running on dedicated infrastructures, storing video in trusted servers owned by systems administrators. Newer installations are usually large scale (commonly hundreds of cameras), heterogeneous and have large differences in resource requirement. [IPVM, 2021].

In this paper the focus is video surveillance systems based on H.264 video cameras, the most prevalent system on the market today. H.264 is a video compression standard based on block-oriented and motion-compensated coding [ITU-T, 2010]. A model of the bandwidth generated, and hence, storage needed, by a H.264 video surveillance camera was presented in [Edpalm et al., 2018; Edpalm et al., 2018]. The model provides an estimate of the bandwidth needs for a H.264 video, given current scene conditions and specific camera parameters. It allows to calculate the long term resource needs for the camera as long as it keeps the current parameters. Anticipating the amount of storage and bandwidth needed by each camera is difficult due to the uniqueness of each scene, camera characteristics and parameters. The amount of storage available is limited and is one of the main cost of running the system [IPVM, 2021]. Furthermore, the cameras compete (or are at the least not explicitly incited to cooperate) for the storage resources available. As a result the system administrators can not trust the devices to provide their real valuation.

This creates a need for strategies to determine the allocation of storage resources that do not rely on trustworthy information being shared between cameras and storage units. We propose the use of auction theory, in particular the Vickrey-Clarke-Groves (VCG) mechanism [Krishna and Perry, 1998], to decide how to allocate these resources. The advantage of VCG auctions is that they provide guarantees, in particular enforcing a fair and envy-free allocation [Pápai, 2003] (an outcome in which each agent does not envy what some other agent has obtained) without requiring control over all devices participating in the auction. Specifically, in this paper we propose a solution to allocate storage resources in a competitive camera system while separating the resource providers (*i.e.*, the storage units) from the resource buyers (*i.e.*, the cameras). The buyers have private information on the amount of resources needed and aim to maximize their valuation (in this case to minimize the compression of their video stream). The storage units enforce the constraint on resource availability by solving a constrained knapsack problem to allocate the resources (see Section 4). This paper focuses on the auction framework and utility determination. The cameras are not explicitly cooperating as the system could be running cameras from different providers which prevents explicit cooperation between devices, and the storage provider acts as a ring buffer as explained in [Martins et al., 2020]. We chose to use auction theory to distribute the available resources in the best way possible without relying on the devices truthfulness.

The contributions of this paper are:

- A game theoretic approach based on VCG auctions for storage allocation in camera systems for video surveillance is proposed.
- A utility measure for camera systems based on the video compression value and its variation is proposed.
- Simulation results of the storage allocated for video content is done to validate the game theoretic proposed solution as well as its system resource cost.

2. Related work

Centralized bandwidth allocation techniques using control theory with maximization of the system-wide visual quality have been proposed in [Seetanadi et al., 2018] and [Silvestre-Blanes et al., 2011]. Second price auctions have been applied to video surveillance systems mostly for specific applications such as area overage [Chong Ding et al., 2012; Konda et al., 2016; Dieber et al., 2011], camera placement [Elhamifar and Vidal, 2009; Ermis et al., 2010] and object tracking [Qureshi and Terzopoulos, 2009; Sankaranarayanan et al., 2008]. Auction theory has been previously used in various computer science applications such as content delivery delay and cashing cost minimization for large mobile networks involving multiple stakeholders as reported in [Li et al., 2016; Ghosh et al., 2004; Pillai and Rao, 2016]. There are also various studies on resource management in cloud computing, mostly focusing on virtual machine resource allocation [Xu and Yu, 2014], some of which are using knapsack optimization to allocate resources, *e.g.*, [Vanderster et al., 2009], or Stackelberg game allocation of CDN resources, *e.g.*, [Li et al., 2016; Hung et al., 2018] or device to device communications, *e.g.*, [Sawyer and Smith, 2019]. Auction theory has also been applied to spectrum sharing in mobile networks [Suris et al., 2007; Cramton et al., 2002] and task allocation to mobile devices such as [Wang et al., 2017] where the VCG mechanism is used to allocate computation tasks to mobile devices or using consensus-based auctions (as explained in [Zlot, 2006]) to ensure consensus between mobile robots, *e.g.*, [Brunet et al., 2008; Choi et al., 2009; Hunt et al., 2014] or [Nanjanath and Gini, 2006]. However, to the authors' best knowledge, auction theory has not been applied to storage allocation for video surveillance systems before.

3. System description

We consider a simplified camera system with one storage unit $\mathcal{P}$ (physically realized as Network Attached Storage, Cloud storage, or some other technique) and C video cameras indexed by i: $\{c^1, c^2, \ldots, c^C\}$. An overview of the system with $C = 4$ is shown in Figure 1. Typically a video surveillance system is owned by a security department, which buys or rents storage from an IT department or a cloud provider at a fixed rate. In our system, viewing quality is most important. The main system goal is to maximize the overall global video quality given the current system constraints, i.e., the running cost and the video storage size.

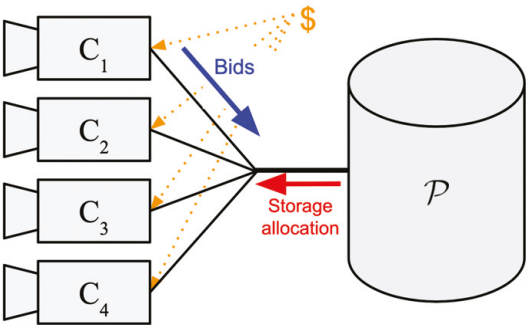


Figure 1. The simplified system considered in this paper.

All cameras can communicate with the seller and can, e.g., be part of the same virtual network. At the beginning of a predefined period k, e.g., an hour, a day, a week, etc, the cameras can buy storage from the seller to save the video they generate during the coming period, using the money at their disposal. If the cameras run out of storage, they need to wait until the next period to buy more. At each period, k, the cameras obtain an amount of money, m, that they can use to buy resources. The amount they receive depends on the cost of running the system. Each camera has a virtual account holding the money it may use. Any remaining money can be saved for future periods. The amount of money available for camera c^i to buy storage at the beginning of each period k is

$$m^i(k) = m^i(k-1) + m,$$

where $m^i(k-1)$ is the money accumulated and available from previous rounds (after all previous payments have been made).

4. Vickrey-Clarke-Groves (VCG) auctions

Vickrey-Clarke-Groves (VCG) is a combinatorial auction mechanism known to yield efficient outcomes, with desirable properties such as incentive compatibility (players best interest is to reveal their true valuation) and individual rationality (players will always benefit from entering the auction) provided the resource allocation is optimal, [Maillé and Tuffin, 2007; Nisan and Ronen, 2007]. VCG auctions apply to any problem where players have a quasi-linear utility function (the compression values being the linear argument).[1] In our case the seller is the storage unit which also acts as the auctioneer. The players are the cameras who want to buy storage, i.e. they are the buyers. The seller is provided a set of proposals, or bids, A, indicating the amount as well as the value that the buyer is willing to pay for this amount. Each buyer sends multiple bids, typically one per compression level. A subset of A, consisting of maximum one bid from each buyer, satisfying the constraints will be selected by the

[1] Quasi-linear utility functions are linear in one argument

seller, leading to an outcome a, which can be considered as an allocation vector. The value that buyer i obtains from this outcome is denoted θ^i. The price p^i that the buyer i pays for the decided outcome a is determined by the VCG mechanism, see (3).

The utility of the buyer is the difference between the willingness to pay θ^i and the price p^i it is charged for it:

$$u^i(a, p^i) = \theta^i(a) - p^i \tag{1}$$

The buyers aim at maximizing this utility.

We assume that buyers are provided with a regular cash inflow in order to be able to buy resources, typically this would be a budget allocated, *e.g.* every auction period, to each camera by the system owner which the buyers can use at their own discretion.

VickreyClarkeGroves auctions work as follows:

1. Each buyer i is asked to reveal his valuation function $\widetilde{\theta}^i$ which indicates how the buyer values each outcome a. The revealed valuation $\widetilde{\theta}^i$ could differ from the real valuation function θ^i if player i is not truthful.
2. The auction mechanism computes an outcome $a^*(\widetilde{\theta})$ that maximizes the declared social welfare, *i.e.,* the sum of revealed valuations $\sum_i \widetilde{\theta}^i$, given the constraints, using 0-1 knapsack optimization, see (4).

$$a^*(\widetilde{\theta}) \in \operatorname*{argmax}_{a \in A} \sum_i \widetilde{\theta}^i(a) \tag{2}$$

3. The price paid by each buyer is given by the loss of declared welfare which the buyer imposes to the others through his presence in the auction, meaning the value that other buyers lose through the difference between the current outcome a^* and an alternate optimal outcome $a' \in a$ without buyer i. For this, we solve one 0-1 knapsack problem per buyer without the items from buyer i present, to find the outcome a' if i was not present, see Eq. 4.

$$p^i = \max_a \sum_{j \neq i} \widetilde{\theta}^j(a') - \sum_{j \neq i} \widetilde{\theta}^j(a^*) \tag{3}$$

   The VCG mechanism is a second-price auction mechanism. Each buyer is declaring its real value and the price paid is the loss of declared welfare which the buyer imposes on the other buyers, as such the buyer will always end up paying less than the declared value. This property is enforced thanks to the secondary knapsack problem in Equation (3), which is done to calculate this loss of welfare to other buyers.

The VCG mechanism verifies three properties [Krishna and Perry, 1998]:

- **Incentive compatibility.** For each user, bidding truthfully (i.e. declaring $\widetilde{\theta}^i = \theta^i$) is a dominant strategy, meaning it is the strategy which will provide the maximum value.
- **Individual rationality.** Each truthful player obtains a non-negative utility, meaning it is advantageous to be truthful (see Eq. 1).
- **Efficiency.** When players bid truthfully, the social welfare, $\sum_i \theta^i$, is maximized.

The auction mechanism decides on the optimal outcome by solving a 0-1 knapsack problem. It is a problem in combinatorial optimization: We pick a set of items (given by the bids), each with a weight and a value. We want to determine the items to include in a collection so that the total weight is less than or equal to a set limit and the total value is as large as possible. In the 0-1 version of this problem each item is indivisible and cannot be picked more than once. Each buyer $i \in [1..C]$ participating in the VCG mechanism provides n bids, indexed by the letter j. The total number of bids for the whole system is thus $N = n \times C$. Each bid contains an item weight w_j^i and its associated declared value v_j^i (weight and value of bid j from buyer i). The declared value is given by the valuation function $v_j^i = \theta(w_j^i)$.

We add one source constraint per buyer to the classical 0-1 knapsack problem, expressing that the optimization is only allowed to select at most one bid j from each buyer i. There is no guarantee that each buyer will have one of its bids accepted whereas no buyer can have more than one of its bids accepted. There will therefore be at most C items selected by this allocation (as there are C buyers). The total weight possible is W and x_j^i indicates if an item j from buyer i is selected ($x_j^i = 1$ if item j from buyer i is selected and 0 otherwise). The modified 0-1 knapsack problem is

$$\begin{aligned} &\text{maximize} && \textstyle\sum_{i=1}^{C}\sum_{j=1}^{n} v_j^i \cdot x_j^i \\ &\text{subject to} && \textstyle\sum_{i=1}^{C}\sum_{j=1}^{n} w_j^i \cdot x_j^i \le W \\ & && \forall i \in [1..C](\textstyle\sum_{j=1}^{n} x_j^i \le 1) \end{aligned} \tag{4}$$

with $v_j^i = \theta^i(w_j^i)$ and $x_j^i \in \{0,1\}$.

5. Estimation of resource needs

5.1 Storage provider

At each auction period $k = \{1,2,...\}$ the storage provider $\mathcal{P}$ sells units of storage (in Gigabyte, Gb). The time between auctions has a duration of T. The total quantity of storage available by storage provider $\mathcal{P}$ is denoted S, the amount of storage for sale at each auction is denoted $s(k)$. $s(k)$ is a subset of S, the total amount of storage available, and the time, R, the data needs to be kept. R is usually determined by the system owner policy. In this paper we define simply $s(k) = \frac{S*T}{R}$, meaning we evenly split the total amount available by the retention time to get the amount to allocation for each auction k of duration T. The storage space allocated to camera c^i is denoted $s^i(k)$. The expected quantities are annotated with a $*$ superscript, *e.g.* the expected allocated storage to camera c^i is denoted $s^{i*}(k)$. Only the storage unit has storage space available, i.e., the cameras are not storage providers.

The storage provider $\mathcal{P}$ acts as a ring buffer, deleting the oldest allocated data in order to accommodate new incoming data. Data stored during the oldest auction period is deleted in order to reuse the allocated storage for the new period k: $\sum_{t=k-R/T}^{k} s(t) \le S$. The storage provider will then assign the storage following the VCG mechanism's

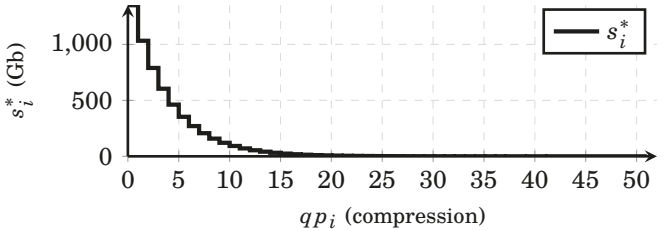


Figure 2. An example of the expected storage as a function of the compression, *i.e.* $s^{i*}(qp)$.

allocation rule, which means that for each auction period k, $\sum_{i=1}^{C} s^i(k) \leq s(k)$ with $s^i(k) \geq 0$ for all cameras c^i.

5.2 Storage buyers (cameras)

In order for a camera to define its storage valuation, θ^i (since VCG is a truthful mechanism, $\widetilde{\theta}^i = \theta^i$), it needs to know how much storage is needed for a video of a certain quality.

For H.264 videos, the compression level is defined by the quantization parameter qp. This value indicates how much data is lost by quantifying the residual data after the transformation step of the encoding process [ITU-T, 2010]. qp is an integer between 0 and 51 ($qp \in \{0..51\}$), 0 being lossless and 51 being the highest compression level. The compression level affects both the memory requirements and the visual quality of the video. The higher the compression, the smaller the amount of storage needed per auction period T will be.

The estimated amount of storage that the camera c^i needs for each H.264 quantization level qp is denoted by $s^{i*}(qp)$ and is calculated using the frame size estimation model provided in [Edpalm et al., 2018]. The model returns the expected amount of storage required for a video with constant scene parameters (motion in the scene, light level, etc.) and settings of the specific camera (frame rate, group of picture length, etc.) for a given qp. An example of this function is shown in Fig.2. The estimation model provided in [Edpalm et al., 2018] is of the form:

$$s^{i*}(qp) = \alpha \times 2^{-qp/6} + \beta \times 5^{-qp/6}, \tag{5}$$

where α and β are positive real numbers defined by a combination of scene and camera parameters. The function $s^*(qp)$ is a positive monotonously decreasing function defined in $\mathbb{R}^+$. This means that it is invertible and we can find $qp^*(s^{i*}) = s^{*-1}(qp)$. In practice the inverse function is found by computing all 52 values of $s^{i*}(qp)$ and inverting the resulting table. Each camera c^i will at the beginning of auction k calculate the $s^{i*}(qp)$ function for all 52 qp values given measurements of the actual scene conditions and, hence, the actual values of the α and β parameters.

6. Valuation of resources

The quality of an H.264 video is directly linked to the compression parameter qp of the video and the variations in qp, see [Cermak et al., 2011; Nemethova et al., 2004; Singh et al., 2012]. In these papers the authors correlated the compression parameter and its variation to the perceived video quality using Mean Opinion Score (MOS) testing, a method to assess video quality by collecting the opinions of participants in a controlled environment. The authors found that the perceived degradation in video quality is strongly noticed at higher compression levels but hardly perceptible at lower levels. Moreover, large variations (jumps) in the compression level are easily noticed by viewers. It means that from a viewer perspective, moving away from the high compression levels is more valuable than getting closer to a very low compression level, i.e., the visual quality gain at low compression levels is hardly noticeable while it would be more noticeable at high compression levels. This is shown in Fig. 4 where the derivative/slope of the red curve is higher towards the higher compression levels. The same applies for compression variations over time, avoiding large changes in compression is more valuable than moving towards no variation.

6.1 Valuation function

In order to minimize the negative effects of high compression and compression jumps as well as take into account their cost to the viewers, we propose a model of the valuation which embodies the desired characteristics, derived from the simple equation of an ellipse. The valuation function θ^i of buyer C^i is defined as

$$\theta^i(qp^i, m^i) = m \cdot \sqrt{1 - \left(\frac{qp^i + \sigma_n(qp^i)}{2 \cdot 51}\right)^2} \tag{6}$$

where m^i is the money available for the camera c^i, qp^i is the compression value corresponding to the received amount s^i, and $\sigma_n(x)$ is the standard deviation of x over the n last periods.

The function embodies our system objective, i.e., to retain video of the highest possible quality in the system given the available money m^i, where quality is measured by the video compression level, qp^i, and how much it varies. The equation of an ellipse has an interesting characteristic around its vertexes. The derivative of the ellipse is low when approaching the co-vertex (low qp and low $\sigma(qp)$), while it is high close to the vertex (high qp and high $\sigma(qp)$). It is valued more (high derivative) to move away from high qp and high $\sigma(qp)$ values (vertex) than it is to get closer to the low qp and low $\sigma(qp)$ values (co-vertex).

6.2 Utility and bid choices

The utility u^i of buyer c^i is then given by

$$u^i(qp^i, m^i, p^i) = \theta^i(qp^i, m^i) - p^i \tag{7}$$

where p^i is the price paid to obtain the amount of storage s^i. The utility is the difference between the value it obtained from the seller and the price it paid to

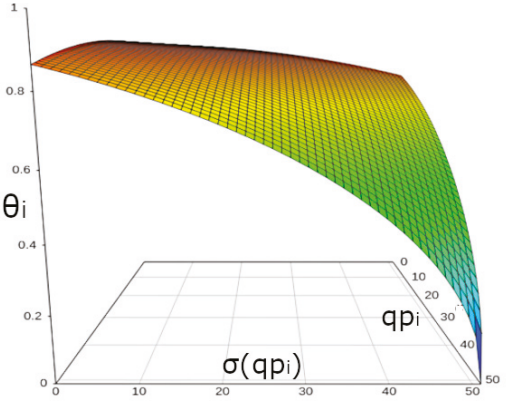


Figure 3. An illustration of how the valuation of resources, θ^i, depends on the compression and its variation.

acquire it. An example of $\theta^i(qp^i, m^i)$ with $m^i = 1$ is shown in Fig. 3 and an example with $m^i = 5$ can be seen in Fig. 4 (blue curve). In Fig. 4 one can also see the predicted amount of storage expected per qp value (black curve) as well as the valuation of each qp values with $m^i = 5$ and fixed θ^i (red curve).

To determine the size of the bids in terms of memory, camera c^i calculates $s^{i*}(qp)$. The value of each quantity in $s^{i*}(qp)$ is the associated valuation $\theta^i(qp^i, m^i)$, One example of how the bids are decided is shown in Fig. 4 (blue curve). One can see that the bids b^i (last plot) are a combination of the expected storage s^{i*} (black curve) and the valuation θ^i (red curve). Each bar in the blue curve represent a bid (an amount of storage s^i and the associated value θ^i).

6.3 Auction steps

The different steps in the VCG mechanism can be summarized as follows:

1. Camera c^i gets a fixed amount of money m for the new period k and adds it to m^i.
2. Camera c^i calculates $s^{i*}(qp)$ and $\theta^i(qp^i, m^i)$ given the actual parameters and sends its 52 bids b^i (one per possible qp): (s^{i*}, θ^{i*}) to the storage provider $\mathcal{P}$.
3. Storage provider $\mathcal{P}$ waits for some time to receive all bids b^i from buyers c^i.
4. Storage provider $\mathcal{P}$ solves a 0-1 knapsack problem with the received bids b^i to find the optimal allocation of available resources, see Section 4.
5. Storage provider $\mathcal{P}$ solves one 0-1 knapsack problem per buyer to calculate the payments of cameras c^i defined by Equation (3) of the VCG mechanism.
6. Storage provider $\mathcal{P}$ sends the allocated storage amount s^i and price p^i to camera c^i.
7. Camera c^i pays the storage provider $\mathcal{P}$ the required amount p^i and starts streaming data up to the allocated storage amount s^i.

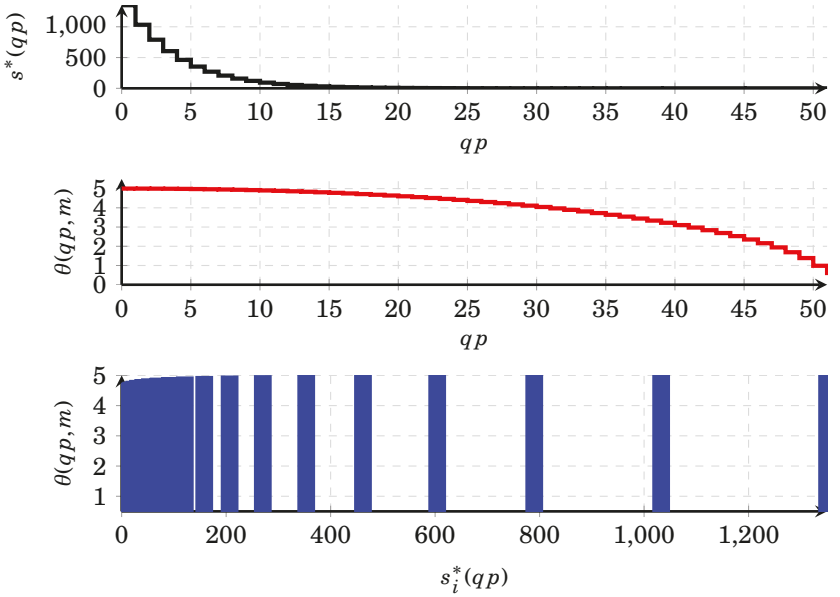


Figure 4. Bid example (with m = 5). The top plot shows the expected storage amount, $s^{i*}(qp)$. The middle plot shows the valuation of the compression, $\theta^i(qp,m)$. The last plot shows the sent bids b^i.

The optimization problem solved is explained in Section 4. The total amount of storage available for sale from the provider $\mathcal{P}$ is S (which is the same as W in Equation 4). x_j^i still indicates if an item j from camera c^i is selected or not ($x_j^i = 1$ if item j from C^i is selected and 0 otherwise).

Each item $j \in [1..n]$ from buyer i has a storage amount s_j^i (denoted w_j^i in Equation 4), and an associated value v_j^i such that $v_j^i = \theta^i(s_j^i)$.

The result of the optimization is the storage amounts s_j^i allocated to each camera c^i, providing the highest possible sum of valuations $\sum_{i=1}^{C} v^i$ depending on the storage resource limitation S of the provider $\mathcal{P}$. As indicated before, $v^i = \theta^i(s^i)$ where $\theta^i(s^i)$ is the declared value of the storage which was given by camera c^i. This value is related to the visual quality of the streamed video as shown in Fig 3.

7. Results

To evaluate the utility function and the use of the VCG mechanism, we implemented a simulator in python and ran multiple simulations with independent players (seller and buyers) communicating via queues.

The 0-1 knapsack problem solving is computationally costly. Because the optimiza-

tion problem is strongly NP-hard, *i.e.* there is no pseudo-polynomial algorithm to solve it [Google, 2021]. This could prevent the system to scale due to time constraints. In this paper we use the Googleő Ortools library (v.9.0.9048) [Google, 2021] which uses the SCIP mixed integer programming solver [SCI, 2021] on an Appleő MacBook Pro with an octacore M1 ARMő-based chip with 3.2 GHz maximum core frequency [Wikipedia, 2021].

The simulations used the Appleő emulator to convert to ARM instructions which slows down the computation but multi-threading was used with up to 10 simultaneous threads. In order to see how long it would take to run a complete VCG auction period, *i.e.*, with $C+1$ knapsack optimizations (C being the number of buyers), we ran 5 simulations with random buyer parameters, 100 auctions each and calculated the mean and standard deviation of the time it took to complete all the required optimizations at each period. The results are shown in Fig. 5. As can be seen, the time to complete the VCG auction follows what appears to be a quadratic function of the number of buyers. As such a system with 50 cameras would solve the assignment of each period in approximately 4.5 seconds, a system with 250 cameras would do so in approximately 3 minutes and a system with 450 cameras would take approximately 12 minutes to do the same task. As long as the auction periods are on the order of an hour, this relatively short time required for computing the allocation is acceptable. If the computation time is a limiting factor one could include a delay of one period in the auction.

For clarity of presentation the following simulations only use four cameras. c^1 is a 4K camera and as such requires the most storage quantity, c^2 is a 1080p camera, c^3 is a 720p camera and c^4 is a 480p camera. The simulation parameters used are: $W = 50$ Gb, $R = 10$ hours, i.e., there is $s(k) = 5$ Gb/hour for sale, and $m = 5\$$ is given to each camera at each period of $k = 10$ hours. Each simulation has fixed camera parameters (resolution, size of the Group of Pictures (GOP), and frame rate) and random video parameters (motion level, light level, noise level, etc). The storage needs of the cameras are determined using the model described in [Edpalm et al., 2018]. The cameras have no explicit incentive to cooperate and try to maximize their own utilities. The values of qp^i in the presented figures are the average values over the period k. Random noise is added to the video frame sizes.

As soon as we consider a competitive system, the buyers, c^i, have no incentive to accommodate other players or provide truthful information if it is not in their favor. For these reasons we compare the VCG mechanism approach (which provides truthfulness as a property) with an equal split of the total resources between the buyers, since this does not assume any information from the cameras. Hence, we compare:

1. Splitting $s(k)$ equally between the buyers c^i.
2. Using the proposed VCG mechanism.

We run two types of simulations for each assignment:

1. The video parameters, change significantly (but in a realistic way) every 10 auction periods.
2. The video parameters are changed randomly every auction period with uniform distributions.

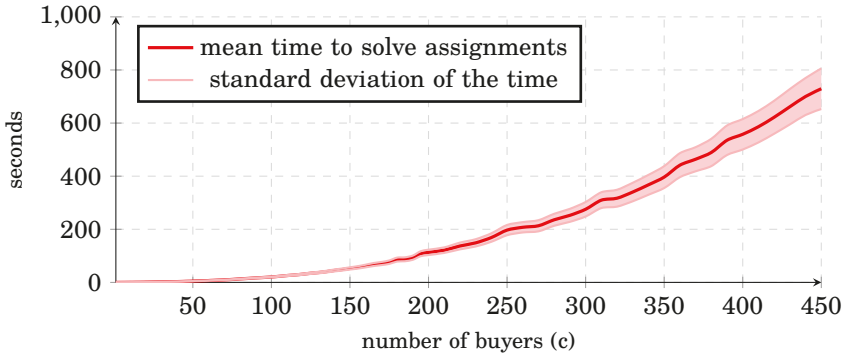


Figure 5. Mean and deviation of the time (in seconds) to solve the assignments per number of buyers C for each period k.

Fig.6 shows the simulation results of the equal allocation (the plots to the left) and VCG allocation (the plots to the right) with low variation of the video parameters (simulation type 1). Fig.7 instead shows the second type of simulation with rapid random changes of the video parameters. The uppermost plots contain the qp^i values of the four cameras, the ones below show the amount of storage s^i allocated and the third plot shows the valuation of the resource acquired, *i.e.,* θ^i, with $m^i = 1$.

We can see in Fig.6 that the VCG approach allocates more storage $s^i(k)$ to c^1 (the 4K camera) which allows c^1 to achieve a lower qp (so better visual quality is perceived by the viewer) by assigning less storage to c^2, c^3 and c^4. The overall visual quality in the VCG allocation case would be more uniform as the camera requiring more storage would be allocated more, which is what the proposed solution aims for.

In Fig.7 we can observe the same behavior as in Fig.6, the VCG mechanism allocates more storage to c^1 allowing it to have a better quality level. Moreover, the variations in qp are also less pronounced than with equal allocation as the cameras c^i react to the parameter changes, modifying the bids b^i accordingly which in turn affects the allocation of storage s^i. By comparing the θ^i curves, we can see the advantage of the VCG mechanism: the video streamed by cameras c^i present closer qp values and less qp deviations which should provide a better system-wide visual quality for the viewer [Cermak et al., 2011; Nemethova et al., 2004; Singh et al., 2012].

One can observe that the VCG approach in this scenario leads to a storage assignment that is very close to a split of the storage resource that is proportional to the camera resolution. However, the latter policy would require the cameras to report this information to the storage, and hence, open up for cameras to provide untruthful information. Using the VCG approach, this assignment is obtained without any need for this information.

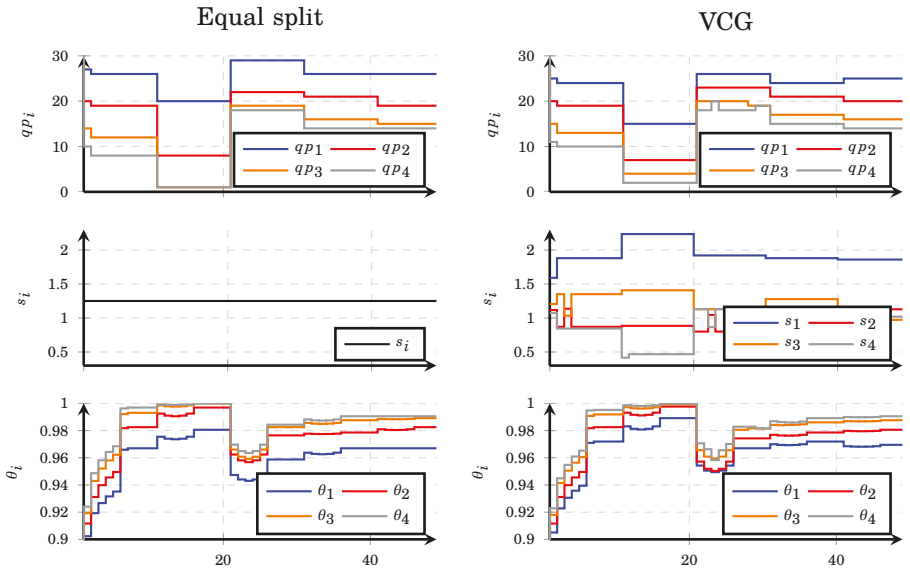


Figure 6. Simulations with 4 buyers and the same amount of money, and low fixed video parameter changes.

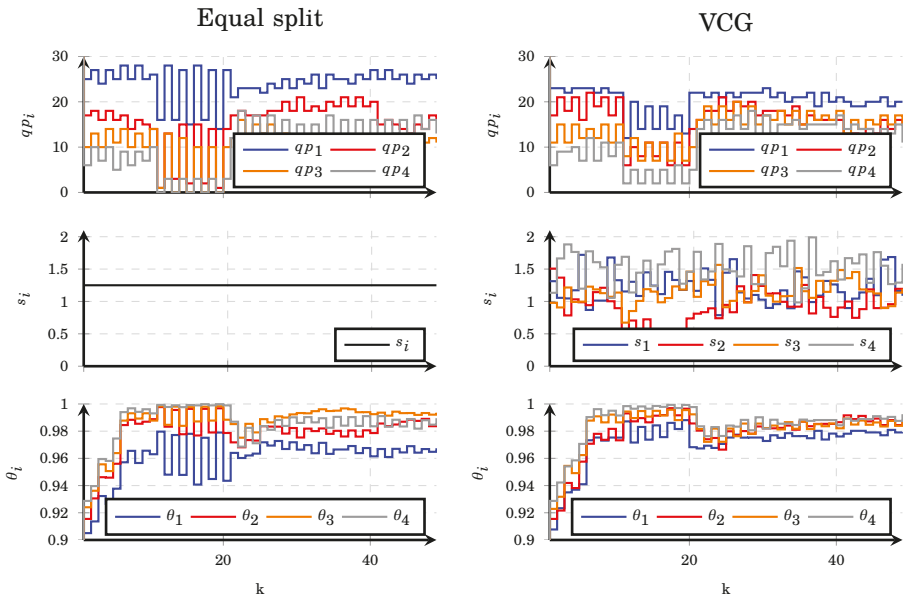


Figure 7. Simulations with 4 buyers and same amount of money, big random video parameter changes.

8. Conclusions and Future Work

In this work we proposed a storage resource allocation method based on the VCG mechanism with the utility derived from the compression level and its variation. The approach requires limited system knowledge and the results in term of system-wide video quality are encouraging. As discussed in [Maillé and Tuffin, 2007], VCG auctions present at least one prohibitive drawback when compared to simpler allocation, as they need computationally intensive NP-complete optimization problems to be solved. The amount of optimizations grows with the number of cameras in the system as each added camera comes with 52 news bids for the optimization problems and one extra optimization for the payment calculation. This still seems to be computationally feasible for the considered systems (hundreds of cameras with allocations every hour or so). The approach has interesting properties (incentive compatibility, individual rationality and efficiency) for systems with competitive players. Thanks to these properties the seller can be unaware of the camera parameters and efficiently allocate the storage based solely on the cameras' declared values. A logical extension of this paper would be to handle multiple storage providers and incorporate learning in the valuation of resources from the cameras. Also, the video quality is here considered to be correlated with the video compression. An alternative approach would be to use an application-specific metric or a recognized quality metric such as the structural similarity index measure (SSIM), peak signal-to-noise ratio (PSNR) or other metrics as described in [Yang, 2007], but at the expense of additional complexity for the cameras.

References

Brunet, L., H.-L. Choi, and J. How (2008). "Consensus-based auction approaches for decentralized task assignment". In: *AIAA guidance, navigation and control conference and exhibit, Honolulu, USA*.

Cermak, G., M. Pinson, and S. Wolf (2011). "The relationship among video quality, screen resolution, and bit rate". *IEEE Transactions on Broadcasting* **57**:2.

Choi, H.-L., L. Brunet, and J. P. How (2009). "Consensus-based decentralized auctions for robust task allocation". *IEEE Transactions on Robotics* **25**:4, pp. 912–926.

Chong Ding, Bi Song, A. Morye, J. A. Farrell, A. K. Roy-Chowdhury, C. Ding, B. Song, A. Morye, J. A. Farrell, and A. K. Roy-Chowdhury (2012). "Collaborative Sensing in a Distributed PTZ Camera Network". *IEEE Transactions on Image Processing* **21**, pp. 3282–3295.

Cramton, P. et al. (2002). "Spectrum auctions". In: vol. 1. Elsevier, Amsterdam, pp. 605–639.

Dieber, B., C. Micheloni, and B. Rinner (2011). "Resource-aware coverage and task assignment in visual sensor networks". *IEEE Transactions on Circuits and Systems for Video Technology* **21**:10, pp. 1424–1437.

Edpalm, V., A. Martins, K.-E. Årzén, and M. Maggio (2018a). *Camera networks dimensioning and scheduling with quasi worst-case transmission time*. Vol. 106. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing.

Edpalm, V., A. Martins, K.-E. Årzén, and M. Maggio (2018b). *H.264 Video Frame Size Estimation*. Technical Report TFRT-7654. Department of Automatic Control, Lund Institute of Technology, Lund University.

Elhamifar, E. and R. Vidal (2009). "Distributed calibration of camera sensor networks". *2009 3rd ACM/IEEE Int. Conference on Distributed Smart Cameras, ICDSC 2009*.

Ermis, E. B., P. Clarot, P.-M. M. Jodoin, and V. Saligrama (2010). "Activity based matching in distributed camera networks". *IEEE Transactions on Image Processing* **19**, pp. 2595–2613.

Ghosh, P., N. Roy, S. K. Das, and K. Basu (2004). "A game theory based pricing strategy for job allocation in mobile grids". In: *18th Int. Parallel and Distributed Processing Symp., 2004. Proceedings.* Pp. 82–.

Google (2021a). *OR-Tools Google Developers Github*. https://google.github.io/or-tools/java/classcom_1_1google_1_1ortools_1_1algorithms_1_1_1KnapsackSolver.html [Accessed: 2021-08-20].

Google (2021b). *OR-Tools Google Developers website*. https://developers.google.com/optimization [Accessed: 2021-03-25].

Hung, Y.-H., C.-Y. Wang, and R.-H. Hwang (2018). "Combinatorial clock auction for live video streaming in mobile edge computing". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops*, pp. 196–201.

Hunt, S., Q. Meng, C. Hinde, and T. Huang (2014). "A consensus-based grouping algorithm for multi-agent cooperative task allocation with complex requirements". *Cognitive computation* **6**:3.

IPVM (2021). *Top 5 problems in video surveillance storage*. URL: https://ipvm.com/reports/problems-video-surveillance-storage (visited on 2021-08-18).

ITU-T (2010). *H.264 standard documentation*. URL: https://www.itu.int/rec/T-REC-H.264 (visited on 2019-06-13).

Konda, K. R., N. Conci, and F. De Natale (2016). "Global coverage maximization in PTZ camera networks based on visual quality assessment". *IEEE Sensors Journal*.

Krishna, V. and M. Perry (1998). *Efficient Mechanism Design*. Tech. rep. https://EconPapers.repec.org/RePEc:wpa:wuwpga:9703010. University Library of Munich, Germany.

Li, J., J. Sun, Y. Qian, F. Shu, M. Xiao, and W. Xiang (2016a). "A commercial video-caching system for small-cell cellular networks using game theory". *IEEE Access* **4**, pp. 7519–7531.

Li, J., H. Chen, Y. Chen, Z. Lin, B. Vucetic, and L. Hanzo (2016b). "Pricing and resource allocation via game theory for a small-cell video caching system". *IEEE Journal on Selected Areas in Communications* **34**:8, pp. 2115–2129.

Maillé, P. and B. Tuffin (2007). "Why VCG auctions can hardly be applied to the pricing of inter-domain and ad hoc networks". *NGI 2007: 3rd EuroNGI Conference on Next Generation Internet Networks*, pp. 36–39.

Martins, A., M. Lindberg, M. Maggio, and K.-E. Årzén (2020). "Control-based resource management for storage of video streams". In: *IFAC World Congress 2020, Berlin*.

Nanjanath, M. and M. L. Gini (2006). "Auctions for task allocation to robots." In: *IAS*.

Nemethova, O., M. Ries, E. Siffel, and M. Rupp (2004). "Quality assessment for H.264 coded low-rate and low-resolution video sequences". In: *Communications, Internet, and Information Technology (CIIT), Saint Thomas, U.S. Virgin Islands*. Citeseer, pp. 508–512.

Nisan, N. and A. Ronen (2007). "Computationally feasible VCG mechanisms". *Journal of Artificial Intelligence Research* **29**, pp. 19–47.

Pápai, S. (2003). "Groves sealed bid auctions of heterogeneous objects with fair prices". *Social choice and Welfare* **20**:3, pp. 371–385.

Pillai, P. S. and S. Rao (2016). "Resource allocation in cloud computing using the uncertainty principle of game theory". *IEEE Systems Journal* **10**:2, pp. 637–648.

Qureshi, F. Z. and D. Terzopoulos (2009). "Planning ahead for PTZ camera assignment and handoff". In: *2009 Third ACM/IEEE Int. Conference on Distributed Smart Cameras (ICDSC)*. IEEE, pp. 1–8.

Sankaranarayanan, A., A. Veeraraghavan, and R. Chellappa (2008). "Object Detection, Tracking and Recognition for Multiple Smart Cameras". *Proceedings of the IEEE* **96**, pp. 1606–1624.

Sawyer, N. and D. B. Smith (2019). "Flexible resource allocation in device-to-device communications using Stackelberg game theory". *IEEE Trans. on Communications* **67**:1.

SCI (2021). *SCIP mixed integer programming solver*. https://www.scipopt.org/ [Accessed: 2021-08-20].

Seetanadi, G. N., L. Oliveira, L. Almeida, K.-E. Årzén, and M. Maggio (2018). "Game-theoretic network bandwidth distribution for self-adaptive cameras". *SIGBED Rev.* **15**:3, pp. 31–36.

Silvestre-Blanes, J., L. Almeida, R. Marau, and P. Pedreiras (2011). "Online qos management for multimedia real-time transmission in industrial networks". *IEEE Transactions on Industrial Electronics* **58**:3, pp. 1061–1071.

Singh, K. D., Y. Hadjadj-Aoul, and G. Rubino (2012). "Quality of experience estimation for adaptive HTTP/TCP video streaming using H.264/AVC". In: *2012 IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, USA*, pp. 127–131.

Suris, J. E., L. A. Dasilva, H. Zhu, and A. B. Mackenzie (2007). "Cooperative game theory for distributed spectrum sharing". In: *2007 IEEE International Conference on Communications, ICC'07, Glasgow, Scotland*, pp. 5282–5287.

Vanderster, D. C., N. J. Dimopoulos, R. Parra-Hernandez, and R. J. Sobie (2009). "Resource allocation on computational grids using a utility model and the knapsack problem". *Future Generation computer systems* **25**:1.

Wang, X., X. Chen, and W. Wu (2017). "Towards truthful auction mechanisms for task assignment in mobile device clouds". In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications, Atlanta, USA*.

Wikipedia (2021). *Apple M1*. https://en.wikipedia.org/wiki/Apple_M1 [Accessed: 2021-08-20].

Xu, X. and H. Yu (2014). "A game theory approach to fair and efficient resource allocation in cloud computing". *Mathematical Problems in Engineering*.

Yang, K.-C. (2007). *Perceptual quality assessment for compressed video*. PhD thesis. UC San Diego.

Zlot, R. M. (2006). *An Auction-Based Approach to Complex Task Allocation for Multirobot Teams*. PhD thesis CMU-RI-TR-06-52. Carnegie Mellon University, Pittsburgh, PA.

Resurshantering i kamerasystem

Alexandre Martins

Institutionen för Reglerteknik

Populärvetenskaplig sammanfattning av doktorsavhandlingen *Resource Management in Distributed Camera Systems*, June 2022. Avhandlingen kan laddas ner från: http://www.control.lth.se/publications

Med utvecklingen av smarta städer och sakernas internet (Internet of Things eller IoT) växer mängden maskin-till-maskin-kommunikation väldigt snabbt. Ett exempel på detta är kamerasystem där små specialiserade smådatorer med bildsensorer kommunicerar via ett nätverk. De analyserar bilder från sensorer, exekverar algoritmer och skickar information och videodata till andra maskiner. Genom att uppskatta och fördela de använda resurserna på ett effektivt sätt kan vi minska resursanvändningen och säkerställa en bättre total servicekvalitet samtidigt som vi håller nere kostnaderna.

Ett typiskt videoövervakningssystem ägs av en säkerhetsavdelning, som köper eller hyr från en central lagringsleverantör P till ett fast pris. Målet är att maximera den totala videokvaliteten givet systembegränsningarna, d.v.s. driftskostnad och videolagringsstorlek. Den lagrade videoövervakningsfilmen bör sparas i några dagar och förstöras efteråt. Varje kamera C i systemet är oberoende och kommunicerar inte med andra kameror, deras enda syfte är att kontinuerligt skicka sin video till en lagringsplats.

Vi fokuserar på tilldelning av två resurser: lagringutrymme och nätverksbandbredd. Dessa resurser är beroende av varandra: tilldelning av den ena påverkar tilldelningen av den andra, eftersom man behöver ha tillräckligt med bandbredd för att kunna lagra en viss mängd data. Vi använder två tekniker som historiskt använts för resurstilldelning: spelteori och reglerteknik. Spelteori har använts vid bandbreddstilldelning i mobila telenätverk på grund av deras stora skala och bristen på förtroende mellan nätverksenheterna. Reglerteknik å andra sidan har använts för mer tidskritiska tillämpningar såsom minnesbandbreddstilldelning eller koordinering av distribuerade system.

Del 1: Reglerteknik tillämpad på kameranätverk.

Vi antar att kamerorna samarbetar och förväntas reagera på lagringsleverantörens P globala feedback angående resurstillgänglighet (Fig. 1.). Diskutrymmet såväl som nätverket mellan kamerorna och lagringsleverantören P är en delad resurs, det är önskvärt att lagra så mycket video som möjligt,

som uppfyller givna kvalitetsbegränsningar, samtidigt som man säkerställer en begränsad fördröjning.

Detta uppnås genom att välja vilken begränsning som är den viktigaste för varje kamera. För en kamera som används för direktvisning är en låg leveransfördröjning avgörande, medan en video dedikerad till lagringsapplikationer kan acceptera en högre fördröjning, men har högre lagringskrav.

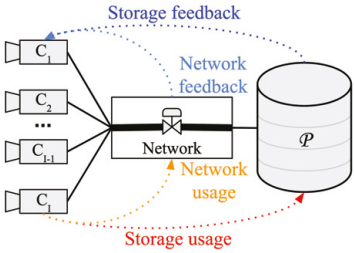


Fig. 1. Kontrollexempel.

För detta fokuserar vi först på tilldelning av diskutrymme och formulerar systemet som ett PI-reglerproblem kombinerat med en metod för att upprätthålla globala resursbegränsningar inspirerad av anti-windup-spårning. Vi formulerar sedan en tredelad modell: kameramodellen, nätverksmodellen och lagringsmodellen. PI-reglering och mid-ranging kombineras för att välja hur mycket av var och en av de delade resurserna som ska tilldelas till varje kamera.

Del 2: Prissättning och auktionsteori tillämpad på kameranätverk.

Här tittar vi istället på ett system där kamerorna C konkurrerar och försöker maximera sin egen videokvalitet oavsett andra resurskonsumenter.

Vi antar att kamerorna vid varje fördefinierad period, t.ex. en timme, en dag eller en vecka, behöver köpa lagringsutrymme från säljaren för att spara videon de genererar (Fig. 2.). Vid varje period fick kamerorna en summa pengar som de kunde använda efter eget gottfinnande för att köpa resurser. Beloppet de fick berodde på kostnaden för att driva systemet.

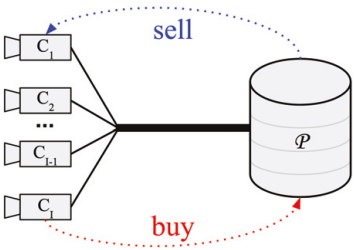


Fig. 2. Auktionsexempel.

Vi använder först principen om prisdiskriminering från mikroekonomi för att tilldela lagringsresurser. Lagringsenheterna tvingade fram begränsning av resurstillgänglighet genom att använda prissättning. Målet för lagringsleverantören P är att konvergera till en optimal och enhetlig kvalitetsnivå för alla kameror i systemet givet de globala resursbegränsningarna. Vi använder därefter auktionsteori, närmare bestämt auktionsmekanismen Vickrey-Clarke-Groves (VCG). Varje kamera C får fortfarande samma summa pengar och skickar till lagringsleverantören P en uppsättning bud som innehåller mängden resurs de vill köpa och kostnaden de är villiga att betala för detta. Lagringsenheterna väljer sedan den optimala tilldelning av resurser och deras kostnader.

Resource Management in Distributed Camera Systems

Alexandre Martins

Automatic control department

Popular science summary of the PhD thesis *Resource Management in Distributed Camera Systems*, June 2022. The thesis may be downloaded from http://www.control.lth.se/publications

With the development of smart cities and the Internet of Things (IoT), the amount of machine to machine communication is fast growing. A good example of this is camera systems where small specialized computers analyze images provided by sensors, run algorithms in quasi real-time and send information and video data to other machines over the network. By estimating and allocating efficiently resources we can reduce the resource waste and ensure a better overall quality of service while keeping the costs of running it low.

A typical video surveillance system is owned by a security department, which buys or rents a central storage provider P at a fixed rate. The main system goal is to maximize the overall global video quality given the system constraints, i.e., the running cost and the video storage size. The stored video surveillance footage should be saved for some days and destroyed afterwards. Each camera C in the system is independent, their sole purpose is to continuously send their video to a storage location.

In this work we focus on allocation of two resources in camera systems: storage and network bandwidth. These resources are dependent: the allocation of one influences the allocation of the other, as one needs to have enough network resources to be able to store an amount of data at the receiver end.

We bridge two techniques historically used for resource allocation: game theory and automatic control. Game theory is used for bandwidth assignment in mobile telecommunication networks due to their large scale and the lack of trust between network the devices. Automatic control on the other hand is used for more specific and time critical applications such as memory bandwidth allocation or coordination of distributed systems.

Part 1: Control theory applied to camera networks.

We assume that the cameras are cooperating and expected to react to the storage provider's P global feedback regarding resource availability (Fig. 1.).

The disk space as well as the network between the cameras and the storage provider P is a shared resource, it is desirable to store as much video

as possible, satisfying given quality constraints, while ensuring a bounded transmission delay.

This is can be achieved by selecting which constraint is the most important for each camera C. 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.

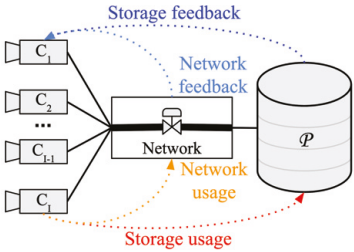


Fig. 1. Control example.

For this, we first focus on disk space allocation and formulate the system as a PI control problem combined with a method for enforcing global resource constraints inspired by anti-windup tracking. We then formulate the model with three parts: the camera model, the network model, and the storage model. PI control with anti-windup tracking and mid-ranging are combined to select how much of each of the shared resources should be allocated to each camera.

Part 2: Pricing and Auction theory applied to camera networks.

Here we consider a competitive setting where each camera C wants to maximize its own video quality regardless of the other resource consumers.

For this we design a framework where at every predefined period, e.g., an hour, a day, etc, the cameras need to buy storage from the seller to save the generated video, using the money at their disposal (Fig. 2.). At each period, the cameras obtain an amount of money that they could use at their discretion to buy resources. The amount they receive depends on the cost of running the system.

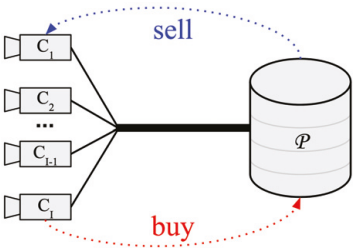


Fig. 2. Auction example.

We first use the price discrimination principle from microeconomics to allocate storage resources. The storage units enforce the constraint on resource availability through the use of pricing. The goal of the storage provider P is to converge to an optimal and uniform quality level for all cameras in the system given the global resource constraints.

We then use auction theory, more precisely the Vickrey-Clarke-Groves (VCG) auction mechanism. Each camera C still receive the same amount of money and send a set of bids containing the amount of resource they desire to acquire and the associated cost they are willing to pay for it to the storage provider P. The storage units then select the optimal allocation of resources and their costs given the provided bids from cameras.