Game-Theoretic Network Bandwidth Distribution for Self-Adaptive Cameras

Nayak Seetanadi, Gautham; Maggio, Martina; Årzén, Karl-Erik; Almeida, Luis; Oliveira, Luis

Published in:
The 15th International Workshop on Real-Time Networks

2017

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

Game-Theoretic Network Bandwidth Distribution for Self-Adaptive Cameras

Gautham Nayak Seetanadi
Department of Automatic Control,
Lund University, Sweden
gautham@control.lth.se

Luis Oliveira
Department of Computer Science,
University of Pittsburgh, USA
loliveira@pitt.edu

Luis Almeida
Instituto de Telecomunicações,
Universidade do Porto, Portugal
lda@fe.up.pt

Karl-Erik Arzén
Department of Automatic Control,
Lund University, Sweden
karlerik@control.lth.se

Martina Maggio
Department of Automatic Control,
Lund University, Sweden
martina@control.lth.se

ABSTRACT

Devices sharing a network compete for bandwidth, being able to transmit only a limited amount of data. This is for example the case with a network of cameras, that should record and transmit video streams to a monitor node for video surveillance. Adaptive cameras can reduce the quality of their video, thereby increasing the frame compression, to limit network congestion. In this paper, we exploit our experience with computing capacity allocation to design and implement a network bandwidth allocation strategy based on game theory, that accommodates multiple adaptive streams with convergence guarantees. We conduct some experiments with our implementation and discuss the results, together with some conclusions and future challenges.

1 INTRODUCTION

Nowadays, networked devices became commonplace, from surveillance cameras to industrial sensors and actuators, or even team of mobile robots. If these devices access the network in an on-demand fashion, sharing bandwidth may result in problems due to network congestion. On the contrary, when the number of devices is unchanged during execution and the communication pattern is very streamlined, network dimensioning should be enough to handle all the simultaneous transmissions in a timely manner.

Unfortunately, there are circumstances in which a proper dimensioning of the network capacity is either impossible, or too expensive. This can for example be the case with a network of cameras. Suppose we have a surveillance camera network, where a certain number of cameras are monitoring a given area. In general, if the designer allocates to each camera the maximum bandwidth they may require, there will be a significant bandwidth over-provisioning. A more efficient approach would be to allocate an average bandwidth so that the cameras can transmit their video streams with an average quality. However, if something is happening in one area – for example a lot of movements are detected by one camera – the bandwidth of the corresponding video stream may be increased, granting it better quality. At the same time, another area may be empty and therefore the corresponding bandwidth can be decreased to accommodate the higher quality stream. The camera network becomes therefore *adaptive* and is able to adjust to the characteristics of the execution environment.

Problem Statement. We consider a system composed of a set of cameras that send video streams to a monitoring node through a shared Ethernet network that supports virtual channels with bandwidth reservations. The cameras must respect their assigned bandwidth, therefore, they run some basic computation on the captured frames to determine the compression level that they should apply. The bandwidth in the network needs to be dynamically allocated at runtime by a monitoring node to accommodate the transient needs of the different cameras. The monitoring node needs to quickly redistribute the available bandwidth introducing as little additional overhead as possible, in particular concerning the transmission of additional information.

Contribution. We propose to achieve the mentioned low-overhead adaptation by decoupling the action of the *resource manager*, in charge of the network bandwidth distribution, and the cameras. Recently, the same strategy has been adopted for CPU allocation [11], where the decoupling of adaptation at the application level and of the adjustment of the scheduling parameters has proven successful. The consequences of allocating CPU can be disruptive for the system but the action itself of allocating CPU itself is is a fairly "safe" operation – with respect to the amount of overhead introduced for the scheduling parameter change – on the contrary changing the channel size has a non-negligible additional overhead in terms of messages exchange, and some associated risks. In fact, if the amount of bandwidth is not enough to stream the frames, every other frame transmission is dropped and a significant reduction in effective video frame rate takes place, with strong negative impact on quality. This paper presents the implementation of a resource manager, allocating network bandwidth to a network of self-adaptive cameras, preserving some guarantees on the transmission of the streams.

Related Work. The topic of self-adaptive cameras has already been investigated for a long time, particularly in the scope of video transmission over the Internet or in local area networks [6, 16, 18]. Research has mainly focused on two complementary issues, video

RTN'17, Dubrovnik, Croatia

This work was supported by the Wallenberg Autonomous Systems Program and the Swedish Research Council (VR) for the projects "Feedback Computing" and "Power and temperature control for large-scale computing infrastructures". Also, this work is a result of the project NanoSTIMA (NORTE-01-0145-FEDER-000016), supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF).

transmission and image compression. The former led to standard protocols such as Real-Time Transport Protocol (RTP), Real-Time Streaming Protocol (RTSP), Session Initiation Protocol (SIP) and their improvements, which measure key network parameters, such as bandwidth usage, packet loss rate, and round-trip delays, to cope with network load conditions, controlling the load submitted to the network [19] or using traffic prioritization with allocation of network resources in the nodes [4].

On the other hand, image and video compression led to standards such as MJPEG, JPEG2000, MPEG-4, H.264 and more recently MPEG-H and H.265 that work by exploring redundant information within each image frame and in sequences of frames. However, these techniques frequently impose strong delays and thus a careful selection must be done for different application domains. While streaming of stored video can tolerate longer delays other domains impose more stringent limitations such as live streaming with augmented reality [15], surveillance [9], industrial supervised multimedia control [16], multimedia embedded systems [14], automated inspection [10] and vehicle navigation [7].

In these cases, image compression is frequently preferred to video compression for the lower latency incurred and lower memory and computing resource requirements. Nevertheless, any compression also incurs variability in transmission frame sizes that further complicates the matching with the instantaneous network conditions and has motivated substantial research into adaptive techniques [14, 16]. However, these works have essentially focused on adapting (single) streams to what the network provides, without protection against mutual interference. This protection can be achieved using network reservations (channels), as with Resource Reservation Protocol (RSVP) or lower layer real-time protocols. However, this has not been common due to high potential for poor network bandwidth efficiency. The work in [17] addressed this problem using adaptive network channels provided by a global network manager that tracks the actual use that each camera is doing of its allocated bandwidth. In this paper we follow this line of work by improving over [17] in the way cameras adapt to their allocated bandwidth, namely using a PI feedback controller, and in the way the manager allocates bandwidth, using a game theoretic approach [11].

2 MODEL

The system is composed of a central node receiving video streams from a set of cameras, $C = \{c_1, \ldots, c_n\}$, with cardinality n, $|C| = n$. The central node also runs a resource manager $\mathcal{M}$, in charge of distributing the available network bandwidth $\mathcal{H}$. Figure 1 shows a system with a node that is in charge of being the network manager and the monitor for the cameras, a switch where bandwidth can be allocated and two cameras sharing the bandwidth.

2.1 The camera

This subsection describes the behavior of a camera c_p with $p \in \{1, \ldots, n\}$. The camera records a stream of frames. Each of these frames is encoded in an image, that is then sent to the central node via the network. The stream of images can be denoted with $I_p = \{i_{p,1}, \ldots, i_{p,m}\}$, where p is the camera identifier and m is the cardinality of the set of images (the longer the system runs, the more images each camera produces). Each element $i_{p,w}$ in the set, $w \in \{1, \ldots, m\}$, has the following characteristics.

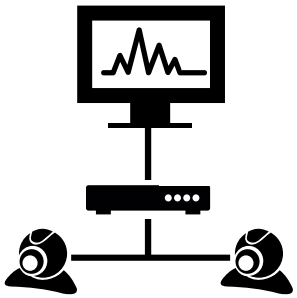

Figure 1: System architecture.

The value of $q_{p,w}$ represents the *quality* used for the frame encoding. The quality is an integer number between 1 and 100, initialized using a parameter $q_{p,0}$, and loosely represents the percentage of information preserved in the encoding. The value of $s_{p,w}$ indicates the *size* of the encoded image. For each of the cameras, depending on the resolution used for the recording and on actual manufacturer parameters, the image size has a maximum and a minimum value, respectively denoted with $s_{p,\max}$ and $s_{p,\min}$, which we assume to be known. Finally, each camera transmits τ_p frames per second, a parameter in our implementation.

The relationship between the quality used for the encoding $q_{p,w}$, which can be changed by the camera, and the size of the resulting frame $s_{p,w}$ is, in principle, rather complex (see [12, 17] for an early exponential model). It depends on many factors, including the complexity of the scene to encode, the sensor used by the camera manufacturer, the amount of light that reaches the sensor. In this work we approximate this relationship using the following affine model

$$s^*_{p,w} = 0.01 \cdot q_{p,w} \cdot s_{p,\max} + \delta s_{p,w}, \tag{1}$$

where $\delta s_{p,w}$ represents a stochastic disturbance on the frame size which can be both positive and negative to capture more difficult or easier scenes to encode. This model is only a coarse-grained approximation of the camera behavior, the idea behind it being that, for control-purposes, the model needs to only capture the trend in the size behavior – increasing quality will more or less linearly increase the frame size, while decreasing quality will more or less linearly decrease the frame size. Linearity is also assumed in the model but not necessarily needed for the controller derivation, as the regulation adapts to the current operating conditions in an adaptive way using a normalized error as seen below. Assuming that a frame size has constraints, we then saturate the result to ensure that the actual size is between the minimum and the maximum size:

$$s_{p,w} = \max\{s_{p,\min}, \min\{s_{p,\max}, s^*_{p,w}\}\}. \tag{2}$$

The camera adapts its behavior, meaning that it automatically changes the quality $q_{p,w}$ to match the amount of network bandwidth that it can use. We denote with $B_{p,w}$ the amount of bandwidth that the p-th camera has available for the transmission of the w-th image at a given frame rate (the channel allows the transmission of a certain number of bytes per frame indicated with $B_{p,w}$). The camera then adjusts its quality parameter using an Adaptive

Proportional and Integral (PI) controller

$$\begin{aligned} e_{p,w} &= \overbrace{\frac{B_{p,w-1} - s_{p,w-1}}{B_{p,w-1}}}^{\text{normalized error}} \\ q^*_{p,w} &= k_p \cdot e_{p,w} + k_i \cdot \sum_{t=1}^{w-1} e_{p,t} \end{aligned} \tag{3}$$

and saturating the result using the minimum and maximum quality values which are 15 and 85 respectively[1],

$$q_{p,w} = \max\{15, \min\{85, q^*_{p,w}\}\}. \tag{4}$$

The gains k_p and k_i are parameters of the controller inside the camera and determine how aggressive the adaptation is. Depending on their values, the system can be analyzed as a dynamical system and standard control theory can be used to prove that the value of $q_{p,w}$ converges to a specific value. Also, the same theory allows to prove that if the conditions of the scene change – for example including more artifacts that makes it more difficult to encode – the quality settles to a new value that allows the transmission of the frame, if such a quality value exists. We also implement an anti-windup mechanism in the controller, a standard practice for PI controllers [1].

2.2 The network manager

To determine how to distribute the network bandwidth we use the approach proposed in [11] for CPU allocation and extend it to handle network bandwidth allocation. The network, particularly the link between the switch and the monitoring station, has a fixed capacity, which we denote $\mathcal{H}$. The network manager $\mathcal{M}$ is in charge of allocating a specific amount of the available network bandwidth to each of the cameras. For every instant of time t in which the network manager is invoked, $\mathcal{M}$ selects a vector $b_{*,w}$, whose elements sum to one.

$$\begin{aligned} \forall t, \mathcal{M} \quad & \text{selects } b_{*,t} = [b_{1,t}, \ldots, b_{n,t}] \\ & \text{such that } \textstyle\sum_{p=1}^{n} b_{p,t} = 1 \end{aligned} \tag{5}$$

This means that each of the elements of the vector determines the fraction of the available bandwidth that is assigned to each video stream. Denoting the actual amount of bandwidth assigned by the resource manager to camera p at time t with $B_{p,t}$, this implies $B_{p,t} = b_{p,t} \cdot \mathcal{H}$. Knowing the frame rate τ_p used by camera p, then the bandwidth allocated by the network manager to that camera at time t in bytes per second ($B_{p,t}$) can be easily converted to the bandwidth (in bytes per frame) allocated to the transmission of frame w, ($B_{p,w}$) as follows: $B_{p,w} = \frac{B_{p,t}}{\tau_p}$.

The network manager is periodically triggered with period $\pi_{\mathcal{M}}$, a parameter in our implementation. Its first invocation, at time 0 equally divides the available bandwidth to the cameras. The following executions, happening at times $\{\pi_{\mathcal{M}}, 2\pi_{\mathcal{M}}, 3\pi_{\mathcal{M}}, \ldots\}$ assign the bandwidth based on the following relationship, from [11], where the index t denotes the current time instant and $t+1$ the following one.

$$b_{p,t+1} = b_{p,t} + \varepsilon \cdot \{-\lambda_{p,t} \cdot f_{p,t} + b_{p,t} \cdot \sum_{i=1}^{n} [\lambda_{i,t} \cdot f_{i,t}]\} \tag{6}$$

[1]Ideally, the quality is a number between 1 and 100, but in our controller we impose saturations that are based on our prior experience with the equipment.

Equation (6) introduces some terms. ε is a small constant and it is used to limit the change in bandwidth that is actuated for every step. Typical values are between 0.1 and 0.6. The choice of a suitable value for ε depends on the trade-off between the responsiveness of the controller (higher values making it converge faster, in principle, but also making it likely to have overshoots) and its robustness to disturbances (lower values delay convergence favoring a more stable behavior in the presence of transient disturbances). $\lambda_{p,t} \in (0, 1)$ is a weight that denotes the fraction of adaptation that should be carried out by the resource manager. A lower $\lambda_{p,t}$ value indicates that the resource manager is less willing to accommodate the needs of the p-th camera. The importance of this value lays in the relative difference between the values assigned to all the cameras. If all the cameras have an equal $\lambda_{p,t}$, the resource manager is not going to favor any of them. If one of the cameras has a higher value with respect to the others, the resource manager is "prioritizing" the needs of that camera over the others, and the changes will favor that specific camera. In the following, we will assume $\lambda_{p,t}$ to not change during execution, and use λ_p instead. A change in the value of λ_p has no impact in our analysis, and can be used to change the resource manager preference during runtime. $f_{p,t}$ is a function that we call the *matching function*, and expresses to what extent the amount of network bandwidth given to the p-th camera at time t is a good fit for the current quality. Denoting with w the index of the last transmitted frame at time t, and with t_w the time of transmission of the w-th frame, f_{p,t_w} determines a match between the quality $q_{p,w}$ and the bandwidth b_{p,t_w} available for the camera when the transmission of the w-th frame happens. For the analysis in [11] to hold (therefore obtaining proof for the properties discussed in Section 2.3), the matching function should satisfy the following properties:

(P1a) $f_{p,t_w} > 0$ if $B_{p,t_w} > s_{p,w}$,
(P1b) $f_{p,t_w} < 0$ if $B_{p,t_w} < s_{p,w}$,
(P1c) $f_{p,t_w} = 0$ if $B_{p,t_w} = s_{p,w}$;
(P2a) $f_{p,t_w} \geq f_{p,t_{w-1}}$ if $q_{p,w} \leq q_{p,w-1}$,
(P2b) $f_{p,t_w} \leq f_{p,t_{w-1}}$ if $q_{p,w} \geq q_{p,w-1}$;
(P3a) $f_{p,t_w} \geq f_{p,t_{w-1}}$ if $b_{p,t_w} \geq b_{p,t_{w-1}}$,
(P3b) $f_{p,t_w} \leq f_{p,t_{w-1}}$ if $b_{p,w} \leq b_{p,t_{w-1}}$.

This basically means that the matching function should be positive if the bandwidth given is abundant, negative if it is insufficient and zero if the match is perfect (P1); that the matching function should increase when the quality is decreased and decrease with increased quality (P2); and, finally, that the matching function increases when more bandwidth is assigned and decreases when bandwidth is removed (P3).

In our implementation we select the matching function to be a normalized version of the mismatch between the bandwidth allocated to the camera and the size of the frame produced by the camera, $e_{p,w}$ in Equation (3):

$$f_{p,t_w} = \frac{B_{p,w} - s_{p,w}}{\underbrace{B_{p,w}}_{B_{p,w} = \frac{B_{p,t}}{\tau_p} = \frac{b_{p,t} \cdot \mathcal{H}}{\tau_p}}}. \tag{7}$$

The so defined matching function automatically satisfies properties (P1a-c) and (P3a-b). If one assumes the disturbance $\delta s_{p,w}$ to be

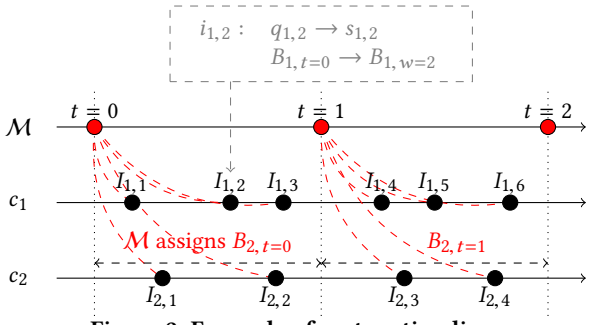

Figure 2: Example of system timeline.

negligible, it is possible to use Equation (1) to verify that properties (P2a) and (P2b) hold. Notice that the matching function corresponds to the normalized error used by the camera controller described in Section 2.1. In the following we will use $f_{p,t}$ to indicate the generic value of the matching function over time and f_{p,t_w} to indicate the precise value of the matching function computed for the frame w transmitted at time t_w. Figure 2 shows a timeline example. At time 0 the network manager decides the values of $b_{1,0}$ and $b_{2,0}$, which in turn assign a value to $B_{1,t=0}$ and $B_{2,t=0}$. Depending on the frame rates τ_1 and τ_2, the cameras have then a value for the amount of bandwidth that each frame should consume. In the case of the first camera, the choice of the resource manager determines $B_{1,w=1}$, $B_{1,w=2}$ and $B_{1,w=3}$. In the case of the second camera, only $B_{2,w=1}$ and $B_{2,w=2}$ are affected. At time $t = 1$ the resource manager chooses a different allocation, affecting the frames in the next interval. For each frame, the cameras determine a quality, that in turn affects the frame size. For example, for $I_{1,2}$, the second image transmitted by the first camera, the quality $q_{1,2}$ determines the frame size $s_{1,2}$. Finally, the following quality $q_{1,3}$ is computed using the difference between the network bandwidth allocated to the frame $B_{1,w=2}$ and the size of the encoded frame $s_{1,2}$.

2.3 System's behavior

From a theoretical perspective, the resource allocation and camera adaptation scheme is not different from the CPU allocation and service level adjustment proposed in [5, 11]. In both cases, there is one entity determining the resource allocation and other entities that can change their resource demands while being cooperative in trying to reach an agreeable resource distribution without unfairly favoring one entity. The behavior of the system has therefore been analyzed and some properties have been proven [5]. Here we only give a brief summary of these properties.

Starvation avoidance. A positive amount of resource is guaranteed for all cameras that have a non-zero weight, $\forall p$ such that $\lambda_p > 0, \forall t, b_{p,t} > 0$.

Balance. The balance property holds in case of overload conditions. The network is overloaded when the capacity $\mathcal{H}$ is not enough to guarantee that all the cameras have a matching function greater or equal to zero $\forall p, \exists i, f_{p,i} \leq 0$, even when $\forall p, q_{p,\min}$. In this case, it is guaranteed that no camera can monopolize the available bandwidth at the expense of the others.

Convergence. The amount of bandwidth allocated to each camera and the streams' quality converge to a stationary point which corresponds to a unique fair resource distribution (a distribution in which the matching function is zero for all the cameras) whenever possible (in non-overload conditions) both in case of synchronous [5, Theorem 4.1] and asynchronous [5, Theorem 4.2] update.

Scalability. The average measured overhead for the computation of the bandwidth distribution in the resource manager is 2μs, for a network of two cameras. Despite this number being small, this is not the reason why we claim this approach has low overhead. One of the reasons why this resource allocation strategy is low-overhead is its linear time complexity. The bandwidth to be allocated can be computed in linear time with respect to the number of cameras, according to Equation (6). This makes the system able to scale to a high number of cameras with limited impact.

3 IMPLEMENTATION AND SETUP

This section describes the underlying protocol which is used to transmit data, together with the acquisition and encoding of images. As shown in Figure 1, the implemented system consists of a network manager and monitor node, together with cameras connected over a Switched Ethernet local area network. The network manager oversees all activity on the network and implements the bandwidth allocation strategy described in Section 2.2. Since it acts as a monitor, it also receives the images transmitted by the connected cameras. It continually monitors the bandwidth consumption and apply bandwidth changes.

OpenCV Each camera p captures an image w and encodes it using the given quality $q_{p,w}$ computed according to Equation (4) in in Section 2.1. The camera than transmits the image to the monitor node. We use OpenCV for image capture and modification, due to its pre-built open source libraries that implement different image processing functionality [3]. Specifically, our implementation uses the `imencode` function, which takes an input image and encodes it using the `jpeg` format and a given compression ratio $c_{p,w}$, that we compute as $c_{p,w} = 100 - q_{p,w}$.

FTT-SE. To dynamically change the amount of bandwidth allocated, we need an underlying architecture that support bandwidth adaptation. For this, we use the Flexible Time Triggered (FTT) scheduling [13], which enforces adaptive hard reservations. In our implementation we use the Switched Ethernet (SE) implementation FTT-SE [12]. We use the asynchronous communication scheme for the FTT-SE setup and select a frame transmission period of 30*ms*, as done in prior work. FTT-SE uses trigger messages from the master (the network manager) to the slaves (the cameras) to change the allocated bandwidth, providing guarantees on minimum bandwidth allocation [2].

Physical Setup. The following section describes experimental results obtained with our implementation. The three physical units used for the experiments form a multiple source single sink architecture. Each unit runs Fedora 21. The first unit has a Intel Core i7-4790, 8 core CPU with 32 GB RAM. It runs the network manager and monitor nodes, implemented as independent processes on the system. The other two units are two cameras, for which we use the commercial off-the-shelf cameras Logitech C270. The first camera was positioned to capture a scene with a lot of artifacts, like fast moving objects. The second camera captures a mostly static scene. We differentiate the scenes to simulate a scenario where cameras have different priorities and needs.

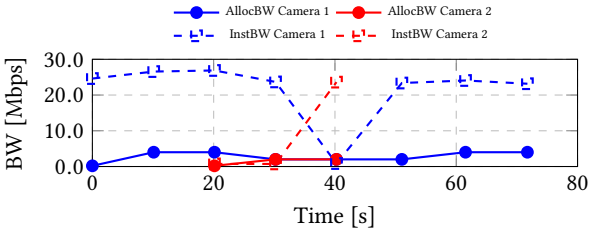

Figure 3: Results with Equal Distribution, without Camera Adaptation – Experiment 1.

Implementation parameters. For the entire infrastructure the implementation parameters are the execution period $\pi_{\mathcal{M}}$, the total available network bandwidth $\mathcal{H}$ and the value of ε. For all of the experiments described in Section 4, $\pi_{\mathcal{M}}$ was set to 300 ms and ε was set to 0.15. In our setup, the available network bandwidth $\mathcal{H}$ is 4 Mbps. We deliberately set a low total bandwidth to stress the system and make sure that adaptation is needed. When two cameras are active, the amount of bandwidth is not enough to transmit the frames, unless the used compression is really high.

Assessment Criteria. We use three different criteria to assess the obtained solutions. The first criterion is the difference between the bandwidth allocated by the resource manager (`AllocBW`) and the one used by the cameras (`InstBW`). The second criteria is called Structural Similarity (SSIM) Index [20]. The SSIM is a metric that represents the information loss from an original image to a transformed one. We use the SSIM to compare the original and encoded image, computing it offline to avoid runtime overhead. Finally, the third assessment criterion is the amount of frames dropped because the allocated bandwidth was not enough to transmit them. The camera captures an image and stores it in the buffer. During transmission if the camera is unable to transmit the whole frame in the allocated bandwidth, it is dropped. Notice that the system does not have enough bandwidth to transmit the full set of frames it records, therefore the optimal percentage of transmitted frames is not 100 (but varies depending on what the network bandwidth allows to achieve).

4 EXPERIMENTAL VALIDATION

We conducted experiments to compare different allocation methodologies (ours and the state-of-the-art solution) and camera adaptation techniques. We recall that there are two adaptation levels: (a) the network manager decides how to distribute bandwidth to the connected cameras, (b) the cameras adapt the encoding process to use the bandwidth.

Experiment 1: need to adjust. In this first experiment, the network manager distributes the bandwidth equally. The cameras do not apply any adaptation mechanism. We use this experiment as a baseline to test the system's operation. Figure 3 shows the experiment results. For the first 20 seconds, there is only one camera in the network, which receives all the bandwidth. A second camera joins the network around 20 seconds after the start and is turned then off when the time is equal to 40 seconds. The manager reacts reducing the amount of bandwidth allocated to first camera and equally distributing network resources to the two cameras. The images produced by the cameras are too big and, in absence of any kind of adaptation, they are often not able to send the data – as reported in the first lines of Tables 1 and 2, only roughly 19% of the frames produced by Camera 1 and 12% of the frames produced by Camera 2 are transmitted.

Table 1: Statistics for Camera 1.

Experiment No	No of Captured Frames	No of Transmitted Frames	Percentage of Transmitted Frames
1	1475	280	18.98
2 [17]	3285	1641	49.95
2 [PI]	3667	1869	50.96
3	1735	805	46.39

Table 2: Statistics for Camera 2.

Experiment No	No of Captured Frames	No of Transmitted Frames	Percentage of Transmitted Frames
1	517	62	11.99
2 [17]	1005	593	59.00
2 [PI]	635	396	62.36
3	634	249	39.27

Experiment 2: comparison with [17]. The next experiment compares two alternatives for the camera adaptation strategies, using the same equal bandwidth allocation described for Experiment 1. As done for Experiment 1, Camera 1 joins the network immediately, while the second one enters at around 40 seconds from the start of the experiment. In both cases, the cameras attempt to match the size of the encoded images to the available bandwidth. Around 85 seconds from the experiment start, Camera 2 is shut down and releases its bandwidth, which is given to Camera 1.

In one case (the left plot in Figure 4), we use the model and adaptation strategy in [17], which fits the Variable Bit Rate (VBR) in the cameras to the given Constant Bit Rate (CBR) channel [8]. In the second case (the right plot in Figure 4), the camera uses the adaptive PI controller described in Section 2 with k_p is set to 10 and k_i is set to 1. We have tuned these parameters for the camera controller empirically according to standard control practice [1]. Compared to the model in [17], our camera is more efficient at using the bandwidth allocated by the network manager. Figure 5 shows the differences in the SSIM per frame. Both the cameras have a SSIM higher than 0. This indicates that quality of images captured in both the cameras is higher with the PI controller compared to the model in [17], making the PI controller a better choice. The amount of frames dropped is similar in both the runs, with the PI model allowing the cameras to transmit slightly more frames (another point in favor) – see Tables 1 and 2.

Experiment 3: the full system. The last experiment incorporates the complete adaptation strategy. The network manager uses the game-theoretic approach to allocate bandwidth to the connected cameras and the cameras use the PI controller to ensure to fully take advantage of the given bandwidth. As a result, the frame that has a lot of artifacts (like Camera 1) and a very time-varying scene is given more network bandwidth. The resulting system is efficient in both allocation and utilization of the allocated bandwidth.

Figure 6 shows the allocation of bandwidth to the two cameras. The network bandwidth starts off by allocating most of the bandwidth available to Camera 1. Around 25 seconds, Camera 2 is

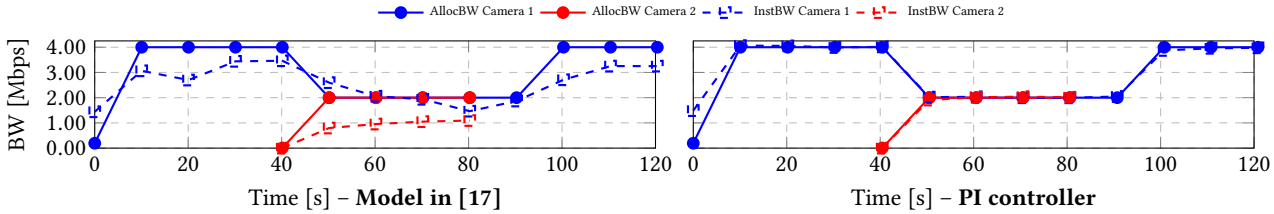

Figure 4: Results with Equal Distribution, and adaptation with [17] (left plots) and PI controller (right plots) – Experiment 2.

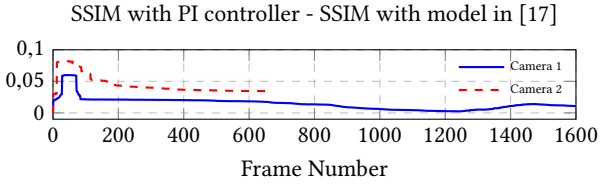

Figure 5: SSIM difference between PI controller and the strategy in [17] – Experiment 2.

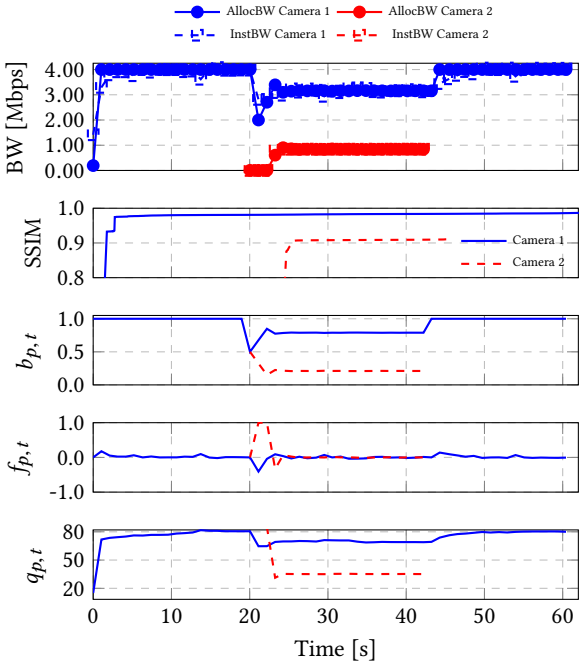

Figure 6: Results with Network manager and PI camera controller – Experiment 3.

introduced. This causes the manager to reconfigure the network and allocate the bandwidth equally. Soon, the manager realizes that Camera 2 does not require as much bandwidth as Camera 1. Thus, the manager adjusts the allocation. Once Camera 2 leaves the network at around 62 seconds, Camera 1 receives the needed additional bandwidth. The Figure also shows the remaining properties of the experiment: the normalized bandwidth $b_{p,t}$ of the camera p at time t that the manager uses to calculate the amount of bandwidth to be allocated, the matching function, $f_{p,t}$ and the quality $q_{p,t}$ set by both cameras.

A negative value of $f_{p,t}$ indicates that the camera is starved and a positive value indicates that the camera has an abundance of bandwidth allocated. The optimal value of $f_{p,t}$ is zero. At every change in the network both the cameras react by changing their qualities and the resource manager by changing the allocation.

5 CONCLUSION

In this paper we have applied a CPU allocation strategy [11] to the problem of network bandwidth allocation with a set of cameras competing for bandwidth. In this paper, we have shown that a resource manager acting periodically in the system is able to achieve some guarantees on convergence, scalability and the general behavior of the system itself.

REFERENCES

[1] K. J. AAström and T. Hägglund. *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, Research Triangle Park, NC, 2 edition, 1995.

[2] L. Almeida. Online qos adaptation with the flexible time-triggered (FTT) communication paradigm. In S. H. S. Insup Lee, Joseph Y-T. Leung, editor, *Handbook of Real-Time and Embedded Systems*. CRC Press, 2007.

[3] G. Bradski. The openCV library. *Doctor Dobbs Journal*, 25(11), 2000.

[4] D. T. Cao, T. H. Nguyen, and L. G. Nguyen. Improving the video transmission quality over ip network. In *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2013.

[5] G. C. Chasparis, M. Maggio, E. Bini, and K.-E. Årzén. Design and implementation of distributed resource management for time-sensitive applications. *Automatica*, 64, 2016.

[6] A. Communication. White paper: Digital video compression: Review of the methodologies and standards to use for video transmission and storage, 2004.

[7] D. A. de Lima and A. C. Victorino. A hybrid controller for vision-based navigation of autonomous vehicles in urban environments. *IEEE Transactions on Intelligent Transportation Systems*, 17(8), Aug 2016.

[8] W. Ding and B. Liu. Rate control of mpeg video coding and recording by rate-quantization modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(1), 1996.

[9] Genetec. White paper: Three simple ways to optimize your bandwidth management in video surveillance, 2010.

[10] A. Kumar. Computer-vision-based fabric defect detection: A survey. *IEEE Transactions on Industrial Electronics*, 55(1), Jan 2008.

[11] M. Maggio, E. Bini, G. Chasparis, and K.-E. Årzén. A game-theoretic resource manager for rt applications. In *Euromicro Conference on Real-Time Systems*, 2013.

[12] R. Marau, L. Almeida, and P. Pedreiras. Enhancing real-time communication over cots Ethernet switches. In *IEEE International Workshop on Factory Communication Systems*, 2006.

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

[14] N. Ramos, D. Panigrahi, and S. Dey. Dynamic adaptation policies to improve quality of service of real-time multimedia applications in ieee 802.11e wlan networks. *Wirel. Netw.*, 13(4), Aug. 2007.

[15] R. Razavi, M. Fleury, and M. Ghanbari. Low-delay video control in a personal area network for augmented reality. *IET Image Processing*, 2(3), 2008.

[16] B. Rinner and W. Wolf. An introduction to distributed smart cameras. *Proceedings of the IEEE*, 96(10), Oct 2008.

[17] J. Silvestre-Blanes, L. Almeida, R. Marau, and P. Pedreiras. Online qos management for multimedia real-time transmission in industrial networks. *IEEE Transactions on Industrial Electronics*, 58(3), March 2011.

[18] B. Vandalore, W.-c. Feng, R. Jain, and S. Fahmy. A survey of application layer techniques for adaptive streaming of multimedia. *Real-Time Imaging*, 7(3), June 2001.

[19] V. Veeraraghavan and S. Weber. Fundamental tradeoffs in distributed algorithms for rate adaptive multimedia streams. *Comput. Netw.*, 52(6), Apr. 2008.

[20] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004.