Exploring Computation and Communication Trade-offs in the Design of Automatic Video Surveillance Networks

Francesco Leonardi Alessandro Pinto Luca P. Carloni Alberto Sangiovanni-Vincentelli

ABSTRACT

Video surveillance is one of the fastest-growing class of networked embedded systems. An increasing number of cameras are networked to support various applications including security in city streets, emergency evacuation in large buildings and direct marketing in department stores. The large number of networked cameras motivates the need for automatic video analysis, which, as of today, relies mostly on centralized computation. Still, trends in embedded computing enable the cost-effective realization of smart camera nodes and, consequently, the distribution of part or all of the computation.

Starting from a particular application of automatic video surveillance for building automation, we derive a system-level model of the main computational tasks that are necessary to process a collection of video streams together with their requirements in terms of computation and communication resources. Then, we define a set of alternative implementation platforms based on a detailed analysis of the possible choices in terms of off-the-shelf components and interconnection network technologies. Finally, we present a methodology and supporting CAD tool that assists us in evaluating alternative partitioning/mapping of the computational tasks onto the various platforms.

1. INTRODUCTION

Video surveillance systems are one of the fastest-growing class of networked embedded systems. These systems are made possible by the combination of several technologies including microelectronics, networking, control systems, and embedded software. The miniaturization of image sensors (cameras) combined with powerful embedded processors allows engineers to build *smart* video-nodes that can process a video stream and transmit it to a central gateway through an interconnection network. Such systems can be used for a variety of applications from security in open spaces such as city streets or university campuses to direct marketing in large department stores.

In this paper we focus on one particular application, i.e. video-based real-time estimation of building occupancy. This application holds the promise of dramatically improving the quality of emergency evacuation procedures in large buildings [1]. Furthermore, to be able to collect real-time information on the occupancy of people in a building would enable major improvements in the operations of HVAC ("heating, ventilating, and air conditioning") systems. In turn, this would allow reducing the building energy costs and raising the level of comfort of its occupants.

The occupancy estimation application requires the deployment of a set of cameras around the building to detect and count people as well as a mechanism to collect and process this information. This high-level specification can lead to many different actual implementations thanks to the large variety of technology solutions, both in terms of embedded computing and networking. In particular, while most video surveillance systems presently rely on centralized computation, the trends in embedded computing are enabling the cost-effective realization of smart video-nodes and, consequently, the distribution of part or all of the computation.

Contributions. In this paper, we first provide a system-level task decomposition of the occupancy estimation application. Then, we present five alternative computation platforms to implement video-nodes of different computational capabilities. This allows us to consider five alternative mappings, each characterized by a different partitioning of the tasks between the video-nodes and the central gateway. Further, each mapping imposes different communication requirements on the interconnect network. We study the communication/computation design trade-offs by developing an analytical model that allows us to estimate the implementation costs across various combinations of the five computation platforms with two network technologies (ARCnet and Ethernet). Finally, we refine the design exploration with the help of a network synthesis algorithm that we have developed within COSI (a software infrastructure for the design of interconnection networks) to find the best implementation matching the physical constraints of a given building.

Related work. A first attempt to count people by means of over-head stereo cameras is presented in [2]. A similar approach is described in detailed by Beymer *et al.* in [3]. It employs a stereo vision sensor, developed by [4], which provides 160×120 pixels frames with 16 depth level. The processing unit, 400-MHz Pentium II executes two main tasks: (1) computing the disparity map and (2) tracking people in each frame. The system showed excellent accuracy in people estimation ($\sim 2\%$ as counting error) with $20 - 25$ frame per second. State-of-the-art sensors [5, 6, 7, 4], which elaborate internally every 3D images, allow reaching much higher frame rates. The utilization of time-of-flight sensors to count people was first proposed by Bevilacqua *et al.* [8] who use a modified version of the algorithms presented in [3] and a 2-GHz Pentium to run detection and tracking algorithms on 64×64 pixel array at 12 frame per second. Snidaro *et al.* presented a set of algorithms to record and process off-line CIF (348×288) images from a color camera [9] but didn't provide a discussion on computational platforms and real-

Francesco Leonardi and Luca P. Carloni are with the Department of Computer Science at Columbia University, New York, NY 10027, {leonardi,luca}@cs.columbia.edu. Alessandro Pinto and Alberto Sangiovanni-Vincentelli are with the EECS Department at UC Berkeley, Berkeley, CA 94720, {apinto,alberto}@eecs.berkeley.edu.

time constraints.

There are many commercial products for both general video-surveillance applications and people counting, but most of these are not based on camera sensors, Axis [10] is one of the leader in video surveillance based on Internet protocol (IP) cameras. The Axis People-Counter comes in the form of a software extension for one of their "Video Server": the AXIS 242S IV. The 242S IV receives one (or more) video streams on a coaxial cable or on a S-video. It can encode the image with MPEG-4 and/or M-JPEG. The output are directed to an Ethernet 10/100 connector. Other ports (RS-232/RS-485) can be used for other purpose (e.g.: controlling a Pan Tilt Zoom (PTZ) camera, connection to other sensors). A dedicated processor (ETRAX 100LX) encodes in real time the video stream. A DSP (TMS320DM642) executes all the other tasks (e.g.: people counting, I/O management, video streaming, etc.). Both the "ETRAX 100LX" and the "TMS320DM642" come with additional 8Mbit Flash and 32MBit memory each.

Biodata [11] offers a scalable solution for counting people. There are two main products: the Video Tally and the Video Turnstile. The first product is a low cost solution able to process up to two-input images. The second product can be extended to allow an indefinite number of camera to be monitored. and it contains a wide set of network interfaces: Ethernet, RS-232, RS-485, WiFi and a modem.

Neuricam [12, 13] (an Eurotech [14] company) offer a people counter based on StereoVision image processing. This solution is completely embedded and it has the option of fusing the measurements of multiple sensors in order to cover wide gate.

LASE PeCo Systems [15] adopts two technologies for sensors: one based on infra-red (IR) and one based on a CCD camera with a resolution of 510×480 pixels.

2. SYSTEM-LEVEL MODELING OF THE OCCUPANCY ESTIMATION APPLICATION

Fig. 1 illustrates the decomposition of the target application, i.e. real-time building occupancy estimation, in a sequence of eight main tasks. This system-level application graph helps us to: (a) identify the key parameters that govern the computation and communication requirements for each stage of such system (b) select the alternative choices in terms of commercial off-the-shelf (COTS) components to build the implementation platform, and (c) complete the design exploration process.

In a centralized implementation each *video-node* contains just the video sensor (the camera) and the hardware that is necessary to interface the node with the network and transmit the video stream to a *network gateway* that acts as *data collector*. Depending on the number of streams to be processed the gateway contains one or more microprocessors running the application software for each of the following tasks and for each stream.

In a distributed implementation, each video-node processes part or all of its video stream by executing a subset of these tasks that are implemented as embedded software running on a microcontroller or a microprocessor. Sometimes it may be more efficient to use a specialized hardware component for a particular task, e.g. H.264 encoding. In general, the number of data that is transferred between one task and the next one decreases as we move down the sequence of tasks. Hence, the more tasks are executed locally on a video-node the lower the bandwidth requirement that a node imposes on the network.

This decomposition is general enough that is applicable to an entire class of distributed video surveillance problems.

Video Sensor. It has been shown that good results for people counting can be achieved even with low resolution gray-scale or 3D image [8, 3, 9, 16, 17]. However, 3D sensors [8, 3, 18] generally provide better performances. For our analysis we assume a 3D video sensor like the one proposed in [3]. The average data bandwidth produced by this sensor is

$$B_{T_{raw}} = NX \cdot NY \cdot FR \cdot PB$$

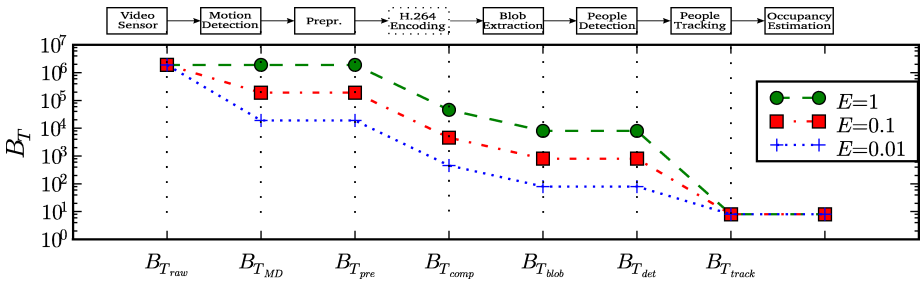
where

- $PB = 4$ is the bit encoding of the 3D image;
- $NX = 160$ is the width of the frame in pixels;
- $NY = 120$ is the height of the frame in pixels;
- $FR = 25fps$ is the number of frames per second.

Computation required at this stage, if present, is minimal. The main workload is to transfer all the pixels from the sensor to memory. This can be performed by simple microcontrollers or DSPs that have dedicated hardware to control image sensors and DMA mechanisms to load pixels in memory without involving a more powerful processor.

Motion Detection. Continuously processing a video stream is a demanding computational task. Designers consider various techniques to save energy and/or transmission bandwidth. A simple idea is based on the observation that the events that need to be detected by the video analysis system do not occur at all times. For instance, in the case of our application, adding a low-cost Pyroelectric InfraRed (PIR) sensor to the video-node makes it possible to detect the presence/absence of a person in the field of view of the camera. In case of absence of people there is no reason to process a video stream and the video-node can remain in a *sleep mode*. Hence, if E denotes the probability of occurrence of an event of interest, then the average output bandwidth of this stage is: $B_{T_{MD}} = B_{T_{raw}} \cdot E$.

Image Preprocessing. In the majority of current techniques a background reference image is first subtracted to each frame. The purpose is to separate the foreground from background. The pixel blobs present inside the foreground are likely to represent people. Especially in case of 2D-images the background can not be considered static. In fact variations in illumination condition and in shadows can significantly change it. This means that if the sensors can not operate in a controlled environment, the background has to be dynamically adjusted over time. Several techniques are described in literature for this purpose. The simplest solution is to refresh periodically the background with a new frame. More sophisticated approaches employ probabilistic evolution model of the background (e.g.: Kalman filter, Gaussian mixture). The more complex the techniques the more time/memory is needed to process it. In case of 3D sensors the background can be more safely considered static: they are quite immune to illumination conditions [19]. The difference image obtained from the previous step shows unavoidably a certain amount of noise which needs to be removed. To this end a set of filters are usually applied. Typology and filter order may vary among different algorithms and authors. Widely used are median, erosion and dilative filters. A "distortion correction" filter has been proposed for 3D images [8, 3]. All these filters do not significantly reduce the image dimension. Therefore, the required bandwidth to transmit the image ($B_{T_{pre}}$) remains the same as in the previous stage, i.e. $B_{T_{pre}} = B_{T_{raw}}$.



Stage	Payload ($Kbps$)
Video Sensor	$B_{T_{raw}} = 1920$
Motion Detection	$B_{T_{md}} = 1920$
Preprocessing	$B_{T_{pre}} = 1920$
H.264 Encoding	$B_{T_{comp}} =\sim 25$
Blob Extraction	$B_{T_{blob}} = 4 - 12$
People Detection	$B_{T_{dct}} = 4 - 8$
People Tracking	$B_{T_{track}} = 0.008$
Occupancy Estimation	$B_{T_{est}} = 0.008$

Figure 1: System-level task decomposition of the building occupancy estimation application highlighting the average data bandwidth transferred between each task.

Image compression. Instead of transmitting the whole raw image, one can insert a stage that compresses the video stream to reduce the required bandwidth of a video-node while experiencing a moderate loss in image quality. Compressing algorithms based on the H.264 standard are currently the most efficient in terms of both compression ratio and image quality. On the other hand, they require a significant amount of computational resources: CPU cycles in case of software implementation, logic gates in case of a hardware implementation based on a dedicated chip or an FPGA design [20]. Since H.264 uses motion-estimation techniques to encode the stream at a fixed image quality, its output data bandwidth $B_{T_{comp}}$ depends on the particular sequence of frames. For our application an average value of about $B_{T_{comp}} = 25Kbps$ can be estimated based on a peak signal-to-noise ratio (PSNR) of $\sim 30\,dB$.

Blob extraction. This stage performs a search on the image to detect the presence of *blobs*, regions of adjacent pixels that have the common property of being brighter than a given level. Depending on the requirement imposed by the subsequent 'people detection' stage, each blob can be expressed as a vector of some scalars. Possible examples are: coordinate of the centroid (x, y, z), average width and height (w, h), average grey (av) level, and the histogram, i.e. a vector whose elements are the number of pixels per each level present in the blob. Other algorithms detect people with only the X and Y projections of each blob [21, 17]. Given that a blob can be encoded with few bytes and that a limited number of blobs is present in each frame then the data bandwidth produced by this stage is on the order of $B_{T_{blob}} = 4 - 12Kbps$.

People detection. This stage processed the blobs received from the previous stage by discarding those that are not compatible with the image of a person and splitting the others based on the number of persons that it is able to detect. In some implementation the people detection is implicit in the blob extraction or in the people tracking [3]. The output bandwidth $B_{T_{det}}$ of this stage is comparable with $B_{T_{blob}}$, even if it is reasonable to assume that the following stage needs just a few features.

If the field of view of one camera is unable to cover all the regions of interest, then a solution is to install multiple sensors. In this case the present stage must properly fuse the video streams produced by different sensors: if N sensors are grouped together, then the overall $B_{T_{det}}$ becomes equal to the sum of the N individual bandwidth values.

People tracking. This stage tracks each person across subsequent frames to establish how many persons have moved from one room to the other (on a frame base) Usually a Kalman filter is the core function at this stage. The resulting output data bandwidth is of the order of few bits, i.e. $B_{T_{track}} \sim 8bps$.

Occupancy estimation. The last task in Fig. 1 can be seen as the front-end of the video surveillance system for our target application. It has to process the output of the sensors and to establish the actual status of the building. An implementation example is presented in [1]. Due to its nature, we have this task always mapped on the processors that are part of the central gateway, which collects the data from all video-nodes.

3. EMBEDDED PLATFORMS

Starting from the application task graph of Fig. 1 we derived a set of alternative embedded platforms to implement the video-nodes for our distributed video surveillance application. The platforms are based on available off-the-shelf components and standard interconnection network technologies. The choice of a particular platform instance corresponds to a possible mapping of the application tasks. This mapping translates into a different set of requirements in terms of both computational power and the communication requirements that the network connecting the video-nodes must satisfy.

Specifically, for the network implementation we considered two field busses that are widely used in building automation and are well suited for this class of application: ARCnet (with EIA-485 as physical layer) and Ethernet.

Figure 2 summarizes the collection of five alternative implementation architectures for the video nodes whose main details we describe in the following sections.

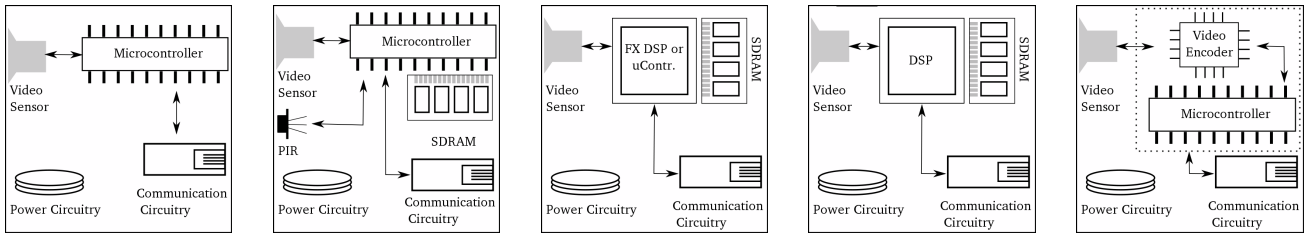
3.1 Raw Transmission Central Computation (RTCC)

The video-node consists of a video sensor and a low-cost micro-controller. Each pixel of each frame is forwarded directly from the sensor to the communication channel. This platform implements just the first task of Fig. 1. The other tasks are executed by the gateway processor.

Embedded Computation. The computation requirements are minimal: the micro-controller implements the entire communication stack and oversees the video sensor. The choice of the particular device is driven by the choice of the network technology because the micro-controller must be able to sustain the necessary communication bandwidth.

Communication. Since the entire frame is transmitted without compression, the network bandwidth requirement is the highest possible for our application, corresponding to the value $B_{T_{raw}} =\sim 2Mbps$. This constraint rules out the effective utilization of ARCnet with the EIA-485 physical layer when the system consists of more than one video-node.

Memory. A frame buffer is not required, because we assume the channel is always available and free. In fact when communications are only in up-link, as in our case study, a switched Ethernet becomes collision free [22]. The necessary memory is integrated in the micro-controller.



(a) RTCC (b) MDCC (c) PDC (d) CDC (e) ETCC

Figure 2: Block diagrams of the video-node embedded platforms described in Section 3.

Embedded Platform	Micro-controller DSPs	Memory		PHY	PHY Chip	PIR ($)	Total ($)	Payload $Kbps$
		FLASH (Mb)	SDRAM (Mb)					
ARCnet enabled								
RTCC	LPC2131 @ 3.84$	-	-	EIA-485	ADM1485 @ 1.98$	-	5.82	1920
DMCC	LPC2131 @ 3.84$	-	64 @ 4$			2	11.82	192
PDC	AT91SAM9260@ 10$	1000 @ 5.64$	64 @ 4$			-	21.62	$4-12$
CDC	ADSP-531@ 15.33$	1000 @ 5.64$	128 @ 8$			-	30.95	0.008
ETCC	i.MX27 @ 15$	1000 @ 5.64$	128 @ 8$			-	30.62	25
Ethernet enabled								
RTCC	LPC2364 @ 6.5$	-	-	Eth 10/100	PC82562EP @ 3.02$	-	9.52	1920
DMCC	LPC2364 @ 6.5$	-	64 @ 4$			2	15.52	192
PDC	AT91SAM9260@ 10$	1000 @ 5.64$	64 @ 4$			-	22.66	$4-12$
CDC	ADSP-536 @ 20.72$	1000 @ 5.64$	128 @ 8$			-	37.38	0.008
ETCC	i.MX27 @ 15$	1000 @ 5.64$	128 @ 8$			-	31.66	25

Table 1: Summary of the embedded platforms components with the associated bill-of-material cost.

3.2 Distributed Motion Detection and Central Computation (DMCC)

With this architecture we move the second task of Fig. 1 into the video node by providing the latter with a motion-detection device such as a simple Pyroelectric Infrared Sensor (PIR). The introduction of this component makes it possible to save dynamically some computational power. Specifically, if the PIR does not detect any movement, then the whole image processing/transmission can be avoided. The cost of this additional sensor and the required circuitry and lenses is about 2 dollars [23].

Embedded Computation. Differently from the first platform this architecture forwards the image to the gateway only when it is needed, e.g. when there is some people movement. The microcontroller, however, must now process also the signal from the PIR.

Communication. The transmission rate is no more constant. The average payload can be estimated as $B_{T_{raw}} \cdot E$ where E is the probability of detecting a movement. This reduction of the required bandwidth allows ARCnet to be taken into consideration for actual installations.

Memory. Because of the change detection stage, a frame buffer is mandatory. In fact, the captured image can be directly transferred to memory. Only when some movements are detected then a certain number of previous and following frames are transmitted to the collector for further processing. In order to allow this behavior, we provide the video-node with $64Mbit$ of SDRAM. This value can be considered as an upper bound. It can be refined and customized on the particular case.

3.3 Partially Distributed Computation (PDC)

This is the most hierarchical among the possible implementation platforms. It requires a video-node with more computational power than in RTCC implementations but still not as much as in the CDC ones. Referring again to the diagram of Fig. 1, the video-node now completes all the tasks up to 'Blob Extraction'. The gateway processor executes the remaining tasks and whenever necessary completes the data fusion of streams arriving from distinct nodes.

Embedded Computation. The video-node is in charge of extracting relevant information from the video stream. It detects blobs in each frame and forwards only some of theirs features to the next processing unit (see Section 2). The nature of these computational tasks demands devices that are able to perform fixed-point arithmetic and run at moderate clock frequency. Our choice is an Atmel micro-controller: the AT91SAM9260, which is built around an ARM9 processor running at $200MHz$ and has native support for Ethernet (MAC) and CAN protocols. Further, as an additional feature it contains an integrated camera interface.

Communication. Since only a few bytes are transmitted for each frame, the resulting output bandwidth is on the order of a few $Kbps$ depending on the number of features required by the subsequent task. All the communication standards mentioned here are viable alternatives.

Memory. With respect to the previous architectures it is necessary to add an external Flash memory to store the embedded software. The video-node now includes $64Mbit$ of SDRAM and $1Gbit$ of Flash.

3.4 Completely Distributed Computation (CDC)

In this case, the computational effort is fully performed by the video-node that must be able to completely analyze the scene and to count people. Only the last task in the diagram of Fig. 1 runs on the central gateway. Each video-node sends to the gateway a few bytes per second. A problem with this solution is the lack of flexibility.

Embedded Computation. Considering that the whole video processing must be executed on the video-node we choose a member of the family of Blackfin DSPs by Analog Devices, i.e. the ADSP-53xx running at 400MHz. Notice that there are many devices with similar computational power but with a different set of integrated I/O interfaces. Thus, the choice of the preferred device depends on the adopted communication channel.

Communication. There is no a priori restriction on the network standard. In fact there are no constraints imposed

by the application: both Ethernet and ARCnet can be considered for the implementation of the CDC solution.

Memory. In order to execute more complex algorithms we must increase the video-node SDRAM to $128Mbit$ while the non-volatile memory remains equal to $1Gbit$.

3.5 Encoded Transmission Central Computation (ETCC)

If one needs to implement a video-surveillance application that requires the collection of some or all the streams from the video-nodes, then other platforms should be considered. A possible solution is the one we describe here that implements the chain of tasks in the graph of Fig. 1 up to the 'H.264 Encoding' stage. In order to overcome the bandwidth issue that is inherent to the RTCC platform an alternative is to encode each video stream. This allows employing a cheaper network infrastructure. A drawback of this choice is that more computational resources must be included in the video-node to encode and decode each stream. In fact, the computational effort required for image compression may be comparable with the one for people counting. On the other hand, building a general-purpose programmable video-node could lead to higher production volumes.

Embedded Computation. The task of encoding a sequence of video frames is pervasive across many applications and many solutions are available. Indeed, there are many products, such as intellectual-property (IP) synthesizable cores for FPGA and/or ASIC, that are actually too powerful for many video-surveillance applications in the sense that are much more expensive than the architectures that we considered so far. As a reference device for this ETCC platform we use Freescale's i.MX27, which consists of one ARM9E core running at 400 MHz and features an integrated H.264 CODEC as well as support for Ethernet MAC 10/100 and CAN protocols.

Communication. The bandwidth of the compressed image stream is of the order of $\sim 25Kbps$, which can be sustained both by ARCnet and by Ethernet.

Memory The i.MX27 needs a FLASH memory to store the embedded software programs and an external memory to execute it. Hence, we need to include $128Mbit$ of SDRAM and $1Gbit$ of FLASH memory in the video-node for ETCC.

4. ANALYTICAL MODEL

We develop a parametric model of the embedded platforms presented in Section 3 and we combine it with a simplified analytical model of the network interconnecting the video-nodes. We consider two candidate network technologies for our application: ARCnet and Ethernet. ARCnet is widely adopted in building automation systems as it offers low installation costs and high flexibility and predictability. Ethernet is a more expensive technology that, however, offers a much wider communication bandwidth, thus making it more attractive for video streaming applications.

4.1 Cost Model for Embedded Platforms

The task graph T depicted in Fig. 1 can be divided in two main parts. Let T_1 be the first part that comprises of the 'Video Sensor' and 'Motion Detection' tasks while let T_2 be the second part that groups together the remaining tasks [1]. The two tasks in T_1 are always active in each video-node since it is always necessary to load the image from the camera sensor into the memory and to establish whether the next task T_2 must be invoked based on the detection of some people movement. Hence, we denote with E the probability of execution of T_2 which depends on the frequency of the observed events (in our example people movements through a door). While T_1 mainly involves algorithms for change detection or object detection, T_2 consists of algorithms for object classification/recognition and tracking, which usually are computationally intense. We capture the different computational loads of the the task groups with the variables L_{T_1} and L_{T_2}, respectively. For the purpose of this analysis, only the relative values of L_{T_1} and L_{T_2} are relevant. Also, since in practical cases $L_{T_1} < L_{T_2}$, we can safely write: $L_{T_1} = 1$ and $L_{T_2} = kL_{T_1}$ with the value of k rounded to an integer number. Next, we examine in details three embedded platforms.

[1]We do not include 'H.264 Encoding' because it is outside the scope of the present model.

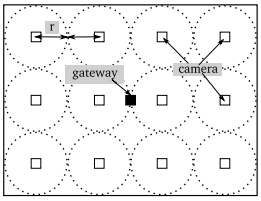


Figure 3: Abstract model of camera deployment.

parameter	definition	value
G	gateway processor	
s	ratio of S_G over S_{CDC}	10
k	ratio of T_2 over T_1	4
E	execution probability of task T_2	0.10
F	frame period	$40ms$
A	area of the building	$2100m^2$
d	density of sensor per m^2	
N	number of sensors	
$A = \frac{N}{d}$	area of the circle	
C_{ARC}	$[\$/m]$: cost per meter of an ARCnet cable	\$8
C_{ETH}	$[\$/m]$: cost per meter of an Ethernet cable	\$12
S_p	number of port in the Ethernet switch	8
C_G	cost of G	\$400
C_{ES_p}	p port switch cost	\$150
C_{AR}	ARCnet router cost	\$760
arc_n	max number of node on an ARCnet chain	8 or 32

Table 2: Analytical model parameters with their reference values.

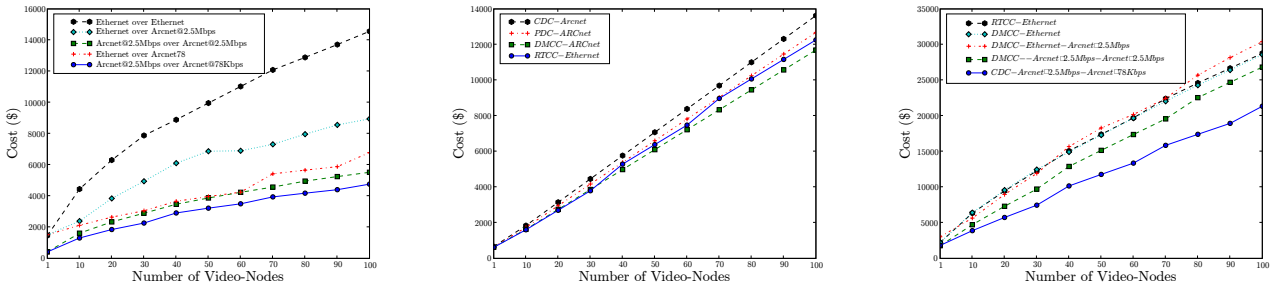
CDC. This implementation is based on fully-distributed computation. Hence, in order to guarantee real-time embedded computing, the processing speed S_{CDC} of the CDC video-node must be sufficient to complete the execution of $T = T_1 + T_2$ within the frame period F, i.e. $S_{CDC} \geq \frac{1+k}{F}$. On the other hand, the gateway processor G must just gather the outputs of the computation in each video camera without doing any major processing. The cost C_{comp}^{CDC} of the hardware to implement the embedded software required by the CDC platform is given by:

$$C_{comp}^{CDC} = C_G + N \cdot C_{CDC_j} \quad (1)$$

where C_G is the cost of the gateway processor, N is the number of cameras, and C_{CDC_j} is the cost of a CDC video-node, which includes an installation cost of about \$100. The index j distinguishes a video-node supporting the ARCnet protocol ($j = 0$) from one supporting the Ethernet protocol ($j = 1$).

RTCC. This implementation is based on fully-centralized computation. Hence no significant processing is embedded in the video-nodes, while the gateway processor G must be able to elaborate a set of streams in a given frame period F. A reasonable assumption is that G has a processing speed S_G that is s times higher than the processing speed S_{CDC} of a CDC video node. Thus, the maximum number P_{RTCC} of streams which can be processed by G is

$$P_{RTCC} = \left\lfloor \frac{s \cdot (1+k)}{1 + k \cdot E} \right\rfloor$$



(a) cabling costs (b) computation costs (c) total costs

Figure 4: Implementation costs (as function of the number of nodes) obtained with the analytical model.

The component cost to implement the computation is:

$$C_{comp}^{RTCC} = N \cdot C_{RTCC_j} + C_G \left\lceil \frac{N}{P_{RTCC}} \right\rceil \qquad (2)$$

DMCC. This implementation is an intermediate choice between CDC and RTCC. It assumes that the task subset T_1 is performed in the video-nodes while T_2 is executed by the gateway processor G only when necessary, i.e. depending on the event probability E. Hence, generally, the number P_{DMCC} of streams that can be processed by G in the unit of time becomes higher than in the RTCC case:

$$P_{DMCC} = \left\lfloor \frac{s \cdot (1+k)}{k \cdot E} \right\rfloor$$

and the overall computation cost is:

$$C_{comp}^{DMCC} = N \cdot C_{DMCC_j} + C_G \cdot \left\lceil \frac{N}{P_{DMCC}} \right\rceil \qquad (3)$$

4.2 Network Cost Models

A significant fraction of the cost of a wired networked embedded system is constituted by the cabling. Many factors influence the cabling cost: e.g. the number and position of the embedded nodes, the physical dimension of the system, the environment constraints and the chosen interconnection technology. We present five high-level analytical models for the following network technologies: ARCnet@78$kbps$, ARCnet@2.5$Mbps$, Ethernet, and two hierarchical heterogeneous networks: Ethernet over ARCnet@78$kbps$ and Ethernet ARCnet@2.5$Mbps$. We assume that N camera sensors are uniformly distributed over a given region (Fig. 3). Let A be the area of the region and $d = \frac{N}{A}$ be the density of the distribution of the sensors in the region. Under these hypothesis the distance between two adjacent sensors can be approximated as $l = 2\sqrt{\frac{1}{\pi d}}$. For simplicity, we place the network gateway at the center of this region.

ARCnet. A network based on ARCnet technology connects all the nodes with one (or more) daisy-chain bus. While various physical layer technologies are compatible with the standard, a common choice is EIA-485. The cabling cost is proportional to the length of the chain:

$$C_{cab}^{A} = (N-1) \cdot l \cdot C_{ARC}$$

ARCnet @78$kbps$ and ARCnet @2.5$Mbps$ use the same wiring medium and connection topology so they have the same cabling cost. An important difference is that on an ARCnet @78$kbps$ chain there can be at most 32 stations, whereas on an ARCnet @2.5$Mbps$ chain this number is reduced to 8. A router must be used to connect multiple chains. In summary the network equipment costs required by the two technologies are respectively:

$$C_{dev}^{A78} = C_G + \left\lceil \frac{N}{32} \right\rceil \cdot C_{AR}$$

$$C_{dev}^{A25} = \left\lceil \frac{N}{8} \right\rceil \cdot C_{AR}$$

Switched Ethernet. A switched Ethernet is a two-tier network. At the first level signals from S_p sensors are merged together by the switch. The second tier connects all the switches to the gateway by means of an additional switch. We assume to install each switch in the middle of its group of sensors. Hence, the Ethernet cable length $l_S(I)$ to connect I sensors to a switch is approximated by:

$$l_S(I) = 2 \cdot l \cdot d_I$$

where d_I is the I-th element of D_I. In case of an 8-port Ethernet switch $D_I = (0, 1, 2, 5, 8, 10, 13, 16)$. The total cable cost in a switched Ethernet is:

$$C_{cab}^{Eth} = [l \cdot l_S(S_p) \cdot \left\lfloor \frac{N}{S_p} \right\rfloor + l \cdot l_S(N \bmod S_p) + \frac{1}{2} \cdot \sqrt{\frac{N}{\pi d}} \cdot \left\lceil \frac{N}{S_p} \right\rceil] \cdot C_{ETH}$$

The first two terms account for the wiring needed by N cameras, whereas the third term represents the cable to connect the switches to the gateway. C_{ETH} summarizes both installation and cable cost per meter. The contribute of switches to the global cost is given by:

$$C_{dev}^{ETH} = \left\lceil \frac{N}{S_p} \right\rceil \cdot C_{ES_p}$$

Hierarchical heterogeneous networks. In these networks the first tier, i.e. the one closer to cameras, is implemented with one technology and the second tier with another. The cabling and component cost of the first-tier networks are the same as above. It is a common design practice to employ a wide bandwidth field bus for the second-tier network (e.g. Ethernet or ARCnet @2.5$Mbps$). In our analysis we take into consideration the following combinations:

- ARCnet @2.5$Mbps$ over ARCnet @78$Kbps$. The cost functions for this network are:

$$C_{cab}^{AII} = (M-1) \cdot l_{II} \cdot C_{ARC} + C_{cab}^{A}$$

$$C_{dev}^{A25II} = \left\lceil \frac{M}{8} \right\rceil \cdot C_{AR} + C_{dev}^{A78}$$

where $M = \lceil \frac{N}{32} \rceil$ is the number of routers instantiated for the first-tier network and $l_{II} = 2\sqrt{\frac{N}{M\pi d}}$ is the distance between two routers.

- ARCnet@2.5$Mbps$ over ARCnet@2.5$Mbps$. Similar to the previous case, except for the value of M which is now $M = \lceil \frac{N}{8} \rceil$.

- Ethernet over ARCnet@78$Kbps$. When Ethernet is the second=tier network we have:

$$C_{cab}^{Eth\_arc78} = [l_{II} \cdot l_S(S_p) \cdot \left\lfloor \frac{M}{S_p} \right\rfloor + l_{II} \cdot l_S(M \bmod S_p) + \frac{1}{2} \cdot \sqrt{\frac{N}{\pi d}} \cdot \left\lceil \frac{M}{S_p} \right\rceil] \cdot C_{ETH} + C_{cab}^{A}$$

$$C_{dev}^{ETH\_arc78} = \left\lceil \frac{M}{S_p} \right\rceil \cdot C_{ES_p} + C_{dev}^{A78}$$

- Ethernet over ARCnet@2.5$Mbps$. Here we have:

$$C_{cab}^{Eth\_arc25} = [l_{II} \cdot l_S(S_p) \cdot \left\lfloor \frac{M}{S_p} \right\rfloor + l_{II} \cdot l_S(M \bmod S_p) + \frac{1}{2} \cdot \sqrt{\frac{N}{\pi d}} \cdot \left\lceil \frac{M}{S_p} \right\rceil] \cdot C_{ETH} + C_{cab}^{A}$$

$$C_{dev}^{Eth\_arc25} = \left\lceil \frac{M}{S_p} \right\rceil \cdot C_{ES_p} + C_{dev}^{A25}$$

where $M = \lceil \frac{N}{8} \rceil$

4.3 Model-Based Estimation of Implementation Costs

We are particularly interested in two aspects of the implementation cost of our target distributed indoor video surveillance system: (1) how the cost changes with the number of video-nodes deployed in a building of and (2) how this cost depends on the platforms of communication and computation components. Using the reference values reported in Table 4.1 we computed model-based estimates of these costs for various implementation technologies and as function of the number of video-nodes in the building.

In particular, Fig. 4(a) reports the estimate of the cabling costs. These show that an Ethernet network is clearly more expensive than any other equivalent network based on the other technologies. The reason is twofold: (1) Ethernet uses a star topology that generally requires more cable than the ARCnet daisy chain busses and (2) the Ethernet cable has an higher installation cost than the ARCnet twisted pair. While a pure ARCnet network appears to be the most cost effective, the curves relative to the hierarchical heterogeneous networks show that combining the two technologies may offer a good compromise between bandwidth and price.

Using Equations (1), (2), and (3) we plotted the computation cost for the platforms CDC, RTCC, and DMCC in Fig. 4(b) [2]. These curves suggest that a centralized architecture is advantageous only for systems having a limited number of video-nodes. In fact, beyond a certain threshold (about 30 sensors in this case) a partial distributed solution becomes more convenient. A completely distributed solution is less efficient from a computation perspective and for this reason more expensive.

Finally, in order to understand the tradeoffs between communication and computation we combined the curves of Fig. 4(a) and Fig. 4(b) to derive those of Fig. 4(c). Generally, this trade-off should reflect the fact that cheaper video-node platforms perform less computation locally and, therefore, impose higher bandwidth constraints on the communication network, which in turn ends up requiring a more expensive communication technology. Still, the estimations based on the given analytical model with the parameters of Table 4.1

[2]The other two platforms are not present because their computational cost is always dominated by these three platforms for the target application.

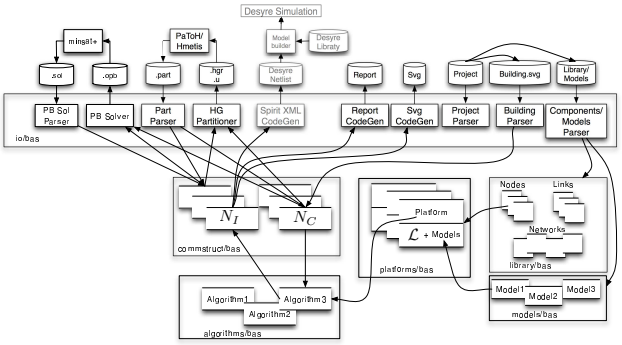


Figure 5: Software organization of COSI

suggest that the cabling costs plays a critical role to tilt the balance in favor of a distributed implementation. In fact, the CDC embedded platform combined with ARCnet @78Kps appears as the most convenient solution regardless of the number of sensors in the system.

5. NETWORK SYNTHESIS

The analytical model presented in Section 4 allows us to explore some of the computation and communication tradeoffs in the design of our video-surveillance application. Some trade-offs are evident: the cost of CDC video-nodes communicating over an ARCnet network is clearly smaller than the cost of RTCC video-nodes communicating over Ethernet. In other cases, however, the cost difference is not as large, which makes design decisions less obvious. For these cases, it is necessary to further refine the models and explore different solutions at a lower level of abstraction. In particular, we want to take into account the physical constraints imposed by the geometry of the particular building which impacts the cabling costs and sometimes even the feasibility of a particular solution. In order to do so, we take advantage of COSI [24, 25], a public-domain design framework that allows researchers and designers to contribute, combine, and compare optimization algorithms, communication protocols, partial designs, and models for the design of interconnection networks.

We first present the design flow that we implemented in COSI and then we provide details about the synthesis algorithm that we develop to generate heterogeneous hierarchical interconnection networks for our target application.

5.1 Communication Synthesis Infrastructure

The Communication Synthesis Infrastructure (COSI) is a framework for modeling and design of interconnection networks. It is based on the principles of platform-based design [26] and it advocates a clear separation of the specification of a communication problem from the possible alternative network implementations. The set of possible implementations, also referred as *the platform*, are implicitly captured by the definition of a library of communication components and composition rules. The problem specification and the communication library are the two main inputs given to a particular communication synthesis algorithm. The algorithm returns the optimal implementation by selecting an element of the platform, called platform instance. The platform instance is obtained through the instancing and the assembling of a set of components that are specified in the communication library.

Figure 5 shows the software organization of COSI. Based on this organization we developed a design flow for the syn-

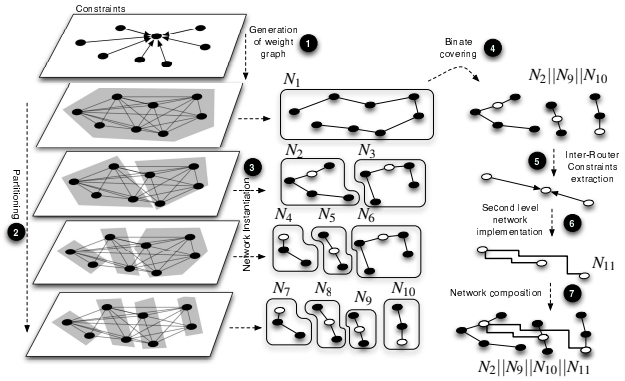


Figure 6: Network synthesis algorithm.

thesis of embedded networks for building automation. Our flow accepts the problem specification in a SVG format that includes: the description of the building floorplan including walls and cable ladders, the fixed location of the sensor nodes (video-nodes) and the candidate locations for the network routers and switches. An internal representation of the specification is generated as a labelled graph N_C that includes the node V_C and the end-to-end communication constraints represented by a set of edges E_C. Each node is annotated with a position in the physical space. Each edge $e(u, v)$ is annotated with a period $t(u, v)$, a message length in number of bits, and a maximum latency constraint $l(u, v)$. The period and the message length are used to derive a bandwidth requirement $b(u, v)$.

The set of the cable ladders constitutes a set of restrictions on the possible layouts for the network wires. In particular, a wire connection between a source node and a destination node must be laid out as much as possible on cable ladders only. Since the wiring cost has a big impact on the total network cost, wiring constraints play an important role in the optimal network design.

The communication library contains the definition of the components that can be used to build the network. Each component is characterized by its interface ports and supported protocols. Performance and cost models are associated to each component to annotate quantities like bandwidth and latency. The models capture the details of the communication protocol like overhead and medium access control. The definition of the components includes the composition rules. These are captured by an operation that defines the performance of a composite network as function of the performance of its components together with a set of constraints that define when a composition is valid. Example of constraints are restrictions on the topology of the network (e.g. a daisy-chain bus, a tree topology, or a star).

The synthesis algorithm returns an optimal implementation N_I that satisfies all end-to-end communication constraints given in the specification while respecting the compositional properties of the network components and the physical constraints imposed by the physical geometry of the building. Notice that there may be problem instances for which a solution does not exist, i.e. using the components in the communication library it is not possible to synthesize a network that satisfies all input constraints. If a solution does exist, COSI returns detailed information on the position of each network components (wires, switches, repeaters,...) as well as a comprehensive report on the cost of the network.

5.2 Synthesis Algorithm

The synthesis algorithm that we developed in COSI is composed of seven steps (Figure 6):

1. The end-to-end communication constraints are first analyzed to derive a complete graph on the set of nodes of the specification that is called the *weight graph*. The edges of the weight graph are labeled with a number that captures the cohesion of two nodes. The weight between two nodes u and v is defined as follows:
$$w(u,v) = \lambda_b b(u,v) + \frac{\lambda_l}{l(u,v)} + \frac{\lambda_d}{d(u,v)}$$
where $b(u, v)$ is the bandwidth requirement, $l(u, v)$ is the latency requirement and $d(u, v)$ is the distance between the nodes (this distance takes into account the wiring constraints).

2. The second step is the partitioning of nodes in subgroups. Partitioning is computed using HMETIS [27]. This is a hyper-graph partitioning tool that divides the nodes of a hyper-graph in groups such that the total weight of the cuts is minimized. The weight graph can be partitioned at different levels of granularity by changing the size of the partitions. Each group of nodes obtained during the partitioning step is a candidate sub-network.

3. After partitioning is completed, each sub-group is processed sequentially to derive a network among its nodes based on the target communication library. The example of Figure 6 assumes the use of ARCnet, which constraints the topology of each sub-network to be a daisy-chain bus. Notice that it is not necessarily the case that a sub-graph can be implemented with a given network technology. This depends on the technology and the constraints of the synthesis problem. The constraints that are taken into account in this step are the length of the wires used to implement the physical connections, the total number of nodes in a subnetwork, the bandwidth constraints, and the latency constraints. During this step, switches are also instantiated for the sub-networks (represented by white dots in Figure 6).

4. A subset of the sub-networks generated in the previous step must be selected for the final implementation. This step entails solving an instance of the binate covering problem. The objective is to select a subset of sub-networks such that each node in the specification is covered by one sub-network, all end-to-end latency constraints are satisfied, and the total network cost is minimized. The selection of one sub-network may prevent the selection of another one because the end-to-end constraints between two nodes in the two subnetworks may be violated. The binate covering problem can be written as follows:

$$\begin{aligned}
\min \quad & \sum_{j=1}^{n} f_j z_j \\
s.t. \quad & \sum_{j=1}^{n} x_{vj} z_j = 1, \forall v \in V_C \quad (1) \\
& \sum_{j=1}^{n} x_{rj} z_j = 1, \forall r \in R \quad (2) \\
& z_i z_j = 0, \forall (u,v) \in E_C, \forall i \neq j \ s.t. \\
& x_{ui} = x_{vj} = 1, l(u, r_i) + l(r_j, v) \geq l(u,v) \quad (3) \\
& z_i, x_{vi}, x_{ri} \in \{0,1\}, \ \forall v \in V_C, \ \forall r \in R, \ \forall i \in [1,n] \quad (4)
\end{aligned}$$

In this formulation, f_j is the cost of the j-th sub-network and z_j is a binary variable that is one if the j-th sub-network is picked in the final solution and zero otherwise. Binary variable x_{vi} is one if node v belongs to the i-th sub-network and zero otherwise. Binary variable x_{ri} is one if switch r serves the i-th sub-network and zero otherwise. Constraint (1) makes sure that a node is covered by exactly one sub-network. Constraint (2) makes sure that each sub-network is covered by exactly one of the candidate switches. Finally, Constraint (3) is the latency constraint. Sub-networks i and j cannot belong to the same solution (i.e. $z_i \cdot z_j = 0$) if the delay from node u to switch r_i (that serves sub-network i) plus the delay from switch r_j (that serves sub-network j) to node v is greater or equal than the maximum delay constraint $l(u, v)$ that is given in the specification.

To solve this binate covering problem we use MiniSat+ [28]. In the example of Figure 6, sub-networks N_2, N_9 and N_{10} are selected and composed.

5. The set of sub-networks selected in Step 4 must be interconnected by a higher-level network linking the switches. Hence a new specification that captures the end-to-end communication requirements among the switches must be generated. This is done by analyzing automatically the solution obtained in the previous step in the context of the specification of the original problem.

6. The higher-level network is synthesized with an algorithm similar to the one described in Step 3 and Step 4. Notice however, that the user may indicate a different platform to implement the higher-level network. Hence, hierarchical heterogeneous networks can be synthesize, e.g. an Ethernet network connecting many ARCnet sub-networks.

7. The final network implementation is derived by composing the results of Step 4 with the higher level network obtained in Step 6.

The algorithm presented in this section is very general and can be used to synthesize networks for a large class of distributed embedded systems using various networking platforms. Notice that only Steps 3 and 6 are technology dependent steps. All other steps do not depend on the network components (i.e. the platform).

6. EXPERIMENTAL RESULTS

In this section we present the results obtained by running COSI to synthesize a set of alternative hierarchical networks for two different buildings (Building A and Building B) and we compare the results with the costs computed by the analytical model and shown in Figure 4. The floor-plan of the two buildings are reported in Figure 7.

We manually distributed a number of video-nodes ranging from 1 up to 60 in proximity of doors and points of access. We also selected 14 possible locations for the installation of switches and routers. These points are uniformly distributed in the building and in proximity of cable ladders. Figure 8 and 9 report the cost of the optimal network synthesized by COSI for five different combinations of network technology as function of the number of video-nodes in the system.

Figure 8 shows some differences with the cost predicted by the analytical model and reported in Figure 4(c). The synthesis result does confirm the clear advantage of using a CDC embedded platform combined with a low-cost network

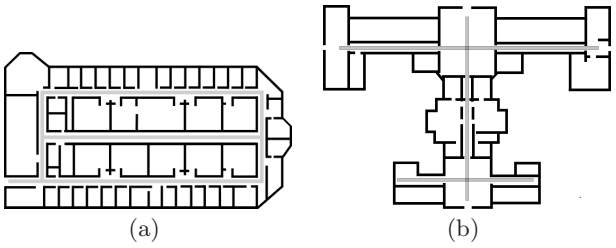


Figure 7: Floor-plan of the two buildings used in our experiments: (a) Building A; (b) Building B.

.

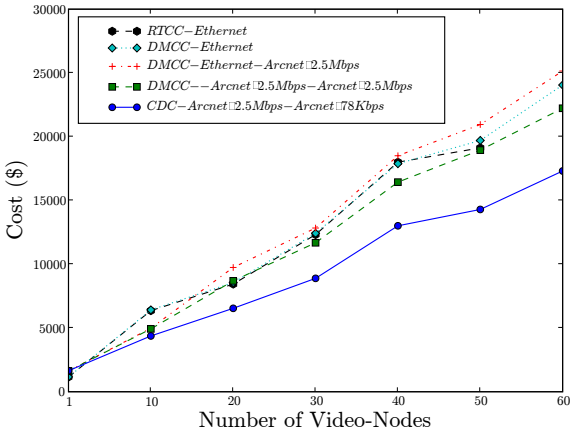


Figure 8: Total cost of computation and communication for Building A of Figure 7(a).

like ARCnet@78Kbps. This may be partially due to the fact that the regular structure of Building A matches fairly well the abstract model of indoor camera deployment of Fig. 3 However, the separation between the DMCC platform with ARCnet and the other implementation platforms is not so clear anymore, especially for systems with a small number of sensors. Also, it is arguable whether a DMCC with ARCnet must always be preferred to the other solutions.

Different considerations must be made when comparing the synthesize results for Building B with the estimates based on the analytical model. The latter is still useful for a coarse-grain evaluation of the trade-offs among solutions. However, using COSI allows us to better determine the impact of combining the DMCC and RTCC computation platforms with different communication technologies. Specifically, for Building B the DMCC computation platform with ARCnet is a more expensive proposition than the RTCC computation platform with Ethernet. This is mainly due to the geometry of the building, which has a different impact on the wiring layout for daisy-chain busses and star topologies. An ARCnet sub-network at 2.5Mbps can connect at most 8 video nodes. An Ethernet switch has at most 8 ports, i.e. an Ethernet sub-network can also connect at most 8 video-nodes. While in Building A a daisy-chain bus is much shorter than an Ethernet star, this is not necessarily the case for Building B. Therefore, Ethernet stars end up costing less than the ARCnet busses. Even the contribution of the central computation is not enough to increase the cost of the RTCC platform and make it comparable to the one of the DMCC platform. The difference in cost can

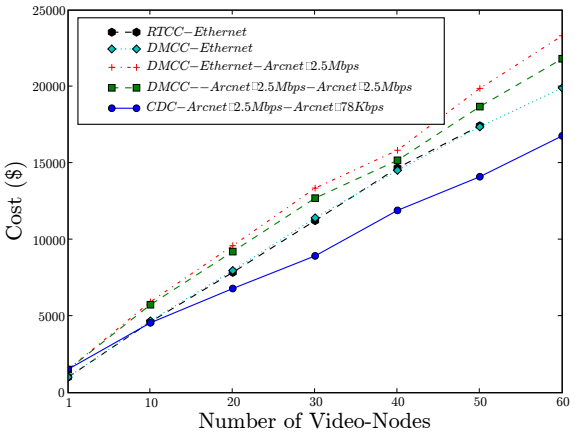


Figure 9: Total cost of computation and communication for the Building B of Figure 7(b).

be as high as 15% for the case of 20 video-nodes. The CDC solution is still the most cost-effective thanks to the higher number of sensors that can be interconnected by the same sub-network.

7. CONCLUSION

Video-nodes, computers, and the network interconnecting them are integral parts of a video surveillance system. The video-node implementation choice impacts the cost of the central computation unit and of the communication network. Therefore, finding an optimal implementation requires to solve a system-level design exploration problem by considering the computation and communication costs trade-offs. When more intelligence is embedded in the video-nodes, less data need to be sent across the network. Hence, more cost-effective network technology can be used with high-end video-nodes. On the other hand, simpler and less expensive video-nodes can be used with a more expensive network that is able to support the bandwidth for the transmission of raw images.

We proposed an analytical model to explore the implementation of the video-nodes and the communication network. We applied this model to the design of a particular video-surveillance application and we found that it is convenient to combine an embedded platform supporting fully-distributed computation with a low-bandwidth network.

While the analytical model can clearly discern the difference between centralized and distributed computation, ranking the cost of less distributed solutions requires a more refined approach that accounts for the impact of the actual building geometry on the network design. Hence we develop an *ad hoc* synthesis algorithm based on the COSI infrastructure for network synthesis and we showed that the cost of the optimal solution depends on the characteristics of the building at hand.

Our future research direction is two-pronged. We plan to tackle the problem of deciding the position and the number of video-nodes given a higher level requirement on the safety and security of the video-surveillance system and a model of the people movement. We will also explore heterogeneous solution that combine distributed and centralized computation to satisfy the more dynamic behavior requirements of other distributed embedded applications.

8. REFERENCES

[1] R. Tomastik, Y. Lin, and A. Banaszuk, "Video-Based Estimation of Building Occupancy During Emergency Egress," *Proc. American Control Conference*, 2008.

[2] K. Terada, D. Yoshida, S. Oe, and J. Yamaguchi, "A method of counting the passing people by using the stereo images," *Proc. of Intl. Conf. Image Processing*, vol. 2, 1999.

[3] D. Beymer, "Person Counting Using Stereo," *Workshop on Human Motion*, p. 127, 2000.

[4] "Videre Design Webpage." [Online]. Available: http://www.videredesign.com/vision/stereo_intro.htm

[5] 3DV Systems, "3DV Systems Webpage." [Online]. Available: http://www.3dvsystems.com/technology/tech.html

[6] Mesa Imaging AG., "Mesa Imaging Webpage." [Online]. Available: http://www.mesa-imaging.ch/

[7] PMD Technologies, "PMD Technologies Webpage." [Online]. Available: http://www.pmdtec.com/e_index.htm

[8] A. Bevilacqua, L. Di Stefano, and P. Azzari, "People Tracking Using a Time-of-Flight Depth Sensor." IEEE Computer Society Washington, DC, USA, 2006.

[9] L. Snidaro, C. Micheloni, and C. Chiavedale, "Video security for ambient intelligence," *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, vol. 35, no. 1, pp. 133–144, 2005.

[10] "Axis Webpage." [Online]. Available: http://www.axis.com/solutions/index.htm

[11] "Biodata Webpage." [Online]. Available: http://www.videoturnstile.com/index.html

[12] B. Crespi and A. Sartori, "Electro-optical device for counting persons, or other, based on stereoscopic vision, and relative method," Patent 20 070 098 253, May, 2007. [Online]. Available: http://www.freepatentsonline.com/20070098253.html

[13] "Neuricam Webpage." [Online]. Available: http://www.neuricam.com/main/product.asp?4M=PCN-1001

[14] "Eurotech Webpage on People Counter." [Online]. Available: http://www.eurotech.com/EN/products.aspx?pg=PCN-1001&pp=Passenger%20Counters&pc=76&pid=9101

[15] "Lase Peco System Webpage." [Online]. Available: http://www.moduloc-peco.com/index.html

[16] T. Teixeira and A. Savvides, "Lightweight People Counting and Localizing in Indoor Spaces Using Camera Sensor Nodes," in *First ACM/IEEE Intl. Conf Distributed Smart Cameras*, 2007, pp. 36–43.

[17] A. Kerhet, M. Magno, F. Leonardi, A. Boni, and L. Benini, "A low-power wireless video sensor node for distributed object detection," *Journal of Real-Time Image Processing*, vol. 2, no. 4, pp. 331–342, 2007.

[18] L. Marcenaro, F. Oberti, G. Foresti, and C. Regazzoni, "Distributed architectures and logical-task decomposition in multimedia surveillance systems," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1419–1440, 2001.

[19] F. Leonardi, D. Covi, and D.Petri, "Metrological Characterisation of a Time of Flight CMOS Range Image Sensor," *Instrumentation and Measurement Technology Conference*, 2008.

[20] Xilinx, "Xilinx H264EncoderBaseline Webpage." [Online]. Available: http://www.xilinx.com/products/ipcenter/H264EncoderBaseline.htm

[21] A. Kerhet, F. Leonardi, A. Boni, P. Lombardo, M. Magno, and L. Benini, "Distributed video surveillance using hardware-friendly sparse large margin classifiers," *Conf. on Advanced Video and Signal Based Surveillance*, pp. 87–92, 2007.

[22] K. Lee and S. Lee, "Performance evaluation of switched Ethernet for real-time industrial communications," *Computer Standards & Interfaces*, vol. 24, no. 5, pp. 411–423, 2002.

[23] Electronics123.com, Inc., "Electronics123.com, Inc." [Online]. Available: http://www.electronics123.com/s.nl/it.A/id.101/.f

[24] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli, "A communication synthesis infrastructure for heterogeneous networked control systems and its application to building automation and control," in *EMSOFT '07: Proceedings of the 7th ACM & IEEE international conference on Embedded software*, New York, NY, USA, 2007, pp. 21–29.

[25] A. Pinto, L. P. Carloni, and A. L. S. Vincentelli, "A methodology and an open software infrastructure for the constraint-driven synthesis of on-chip communications," University of California, Berkeley, Technical Report UCB/EECS-2007-130, November 2007.

[26] A. Sangiovanni-Vincentelli, "Defining platform-based design," *EEDesign of EETimes*, February 2002.

[27] G. Karypis and V. Kumar, "hmetis 1.5: A hypergraph partitioning package," Department of Computer Science, University of Minnesota, Tech. Rep., 1998.

[28] N. S. Niklas Een, "Translating pseudo-boolean constraints into sat," *Proceedings of JSAT*, 2006.