A Case Study in Distributed Deployment of Embedded Software for Camera Networks

Francesco Leonardi
Dept. of Computer Science
Columbia University, New York, NY
francesco.leonardi@ieee.org

Alessandro Pinto
United Technologies Research Center
United Technologies, Hartford, CT
PintoA@utrc.utc.com

Luca P. Carloni
Dept. of Computer Science
Columbia University, New York, NY
luca@cs.columbia.edu

***Abstract*—We present an embedded software application for the real-time estimation of building occupancy using a network of video cameras. We analyze a series of alternative decompositions of the main application tasks and profile each of them by running the corresponding embedded software on three different processors. Based on the profiling measures, we build various alternative embedded platforms by combining different embedded processors, memory modules and network interfaces. In particular, we consider the choice of two possible network technologies: ARCnet and Ethernet. After deriving an analytical model of the network costs, we use it to complete an exploration of the design space as we scale the number of video cameras in an hypothetical building. We compare our results with those obtained for two real buildings of different characteristics. We conclude discussing the results of our case study in the broader context of other camera-network applications.**

I. Introduction

Video surveillance systems are a fast-growing class of networked embedded systems [1], [2], [3]. 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. We study *video-based real-time estimation of building occupancy* as the representative of a large class of camera network applications. This application holds the promise of dramatically improving the quality of emergency-evacuation procedures in large buildings [4]. Also, to collect real-time information on the people occupancy in a building may enable major improvements in the operations of HVAC ("heating, ventilating, and air conditioning") systems that translates into a reduction of building energy costs and higher comfort levels for its occupants.

Building occupancy estimation 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 the information from each camera. This high-level specification can lead to many different actual implementations thanks to the large variety of technology solutions that are available both in terms of embedded computing and networking. In fact, there are companies offering systems that rely on centralized computation [5], [6], [7] while others leverage the trends in embedded computing to propose the cost-effective realization of smart video-nodes and, consequently, the distribution of part (or all) of the processing tasks [8], [9].

We developed the embedded software for our target application and profiled several alternative compositions of its main tasks by collecting measures on three processor architectures: ARM 9, PowerPC, and Core 2 Duo. Based on this analysis we derived five candidate video-node platforms that have different computation/cost characteristics. We then considered various alternative mappings of the application tasks on these platforms together with the communication requirements that they impose on the interconnection network design. To analyze the communication/computation design trade-offs we built 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) as we scale up the number of cameras in the network. These estimates are compared with results obtained for two real buildings before presenting some concluding remarks based on our case study.

II. The Application

We developed the embedded software for real-time building-occupancy estimation by leveraging several results published in recent years. In particular, while some researchers report good results with the use of low-resolution gray-scale or 3D images [10], [11] 3D video sensors generally provide better performance [10], [11], [12]. We chose a 3D sensor that produces a stream of images of 160x120 4-bit pixels at 25 frames per second [11]. As in [10], [11], [13] we developed our application in a modular fashion based on a static dataflow model of computation: each video stream is processed by a sequence of tasks $AS = \{A_{VS}, A_{MD}, A_1, A_2, A_3, A_4, A_5, A_{OC}\}$, which perform the following computations (Fig. 1):

- *Video Sensing (A_{VS}).* This task controls the video sensor and loads the frames into a memory buffer.
- *Motion Detection (A_{MD}).* Since processing/transmitting a video stream requires a fair amount of computations/bandwidth, it is often convenient to equip each camera with a low-cost Pyroelectric Infrared Sensor (PIR) device that performs presence/motion detection and triggers the execution of the subsequent tasks by forwarding only those frames that have possible people presence/motion.
- *Preprocessing (A_1).* This task subtracts a reference background frame from each raw frame and then applies basic morphological filters (erosion and dilation) to improve the

978-3-9810801-5-5/DATE09 © 2009 EDAA

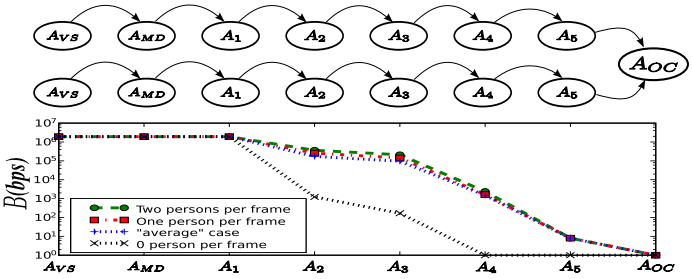


Fig. 1. Inter-task bandwidth requirements for the case study application.

image quality for the subsequent tasks. This module produces the same amount of data than it receives as input.

• *Blob Extraction (A_2).* This task identifies contiguous pixel regions (*blobs*) of a frame that are brighter than the background by performing three sub-tasks: a) scan the image and label non-background pixels; b) cluster contiguous labelled pixels; c) produce the data representation for each blob that holds its pixels and the coordinates of its bounding box.

• *People Detection (A_3).* This task processes the blobs of a frame to establish if it represents a person. It performs three sub-tasks: a) discard those blobs that are too small to represent a person; b) merge "close" blobs as they likely refer to the same object; c) repeat a "local" blob extraction to detected brighter regions for those blobs that are too large for a person image. The output is the set of blobs recognized as persons.

• *People Tracking (A_4).* This task tracks the detected people across frames by maintaining a list of their centers of mass. For each frame, it forms a set of tracked people, each represented with two variables: the current position of the center of mass and the location where the person was first detected.

• *People Counting (A_5).* This task determines when a tracked person crosses an hypothetical threshold between two building areas and updates one of two counters depending on the person direction (in or out).

• *Occupancy Estimation (A_{OC}).* This task collects the processed information for each video-stream in the network and estimates the occupancy for each building area. It is typically executed on a dedicated processor that acts as a gateway for the network of cameras.

III. Embedded Software Profiling

To have a better understanding of the design space for the target application we profiled our software implementation on two widely-adopted embedded processors (Table III): ARM 9, which represents a class of many low-cost micro-controllers used in smart video-nodes, and PowerPC, which represents the set of mid-range CPUs suitable to process multiple video streams. As a reference point, we also profiled our software with a Core 2 Duo.

In general, there are many possible mappings of the application tasks on a target distributed embedded system. Our application has some *radial symmetry* properties. The same sequence of tasks, described in Section II, must be executed for the video-stream produced by each camera in the network and the people counting data must be collected and processed by a gateway processor (Fig. 1-top). Tasks A_{VS} and A_{MD} must be implemented in each video sensor node, while a single task A_{OC} must run on a gateway processor. Hence, it is natural to limit our analysis to all possible *task partitions* P_{ij} that represent ordered sequences of data-dependent tasks $A_i, A_{i+1}, \ldots, A_{j-1}, A_j$ where $i, j \in [1, 5]$ and $i \leq j$.

Actor	Person per frame			
	Average case	0	1	2
$B(A_{VS})$	1920000	1920000	1920000	1920000
$B(A_{MD})$	1920000	1920000	1920000	1920000
$B(A_1)$	1920000	1920000	1920000	1920000
$B(A_2)$	171940	1229	256409	364656
$B(A_3)$	95964	168	143498	209794
$B(A_4)$	1482	0	1625	2386
$B(A_5)$	200	0	400	400
$B(A_{OC})$	0	0	0	0

Table I: Average bit-rate on 1000 frames for four different scenarios.

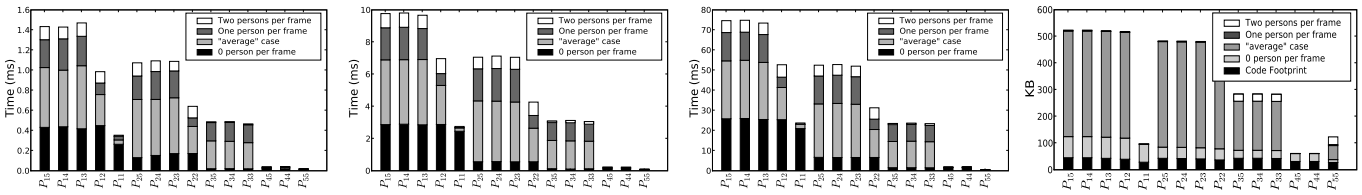
Processor	Speed (MHz)	L1 cache (KB)	L2 cache (MB)	System	GCC version
Core 2 Duo	2000	$32 + 32$	4	HP DV2000	4.2.3
PowerPC750	350	$32 + 32$	1	Macintosh G3	4.1.0
ARM 926EJ	200	$32 + 32$	none	WD MyBook	3.4.2

Table II: Embedded software profiling set-up.

For each partition P_{ij} we measured: the execution time, i.e the time each task partition needs to process a frame, the instruction-memory size (i.e the footprint of the program), the peak data-memory size (heap + stack) and the average output payload per frame (the number of bits that must be transferred to the next task in the sequence without considering any communication protocol overhead). Since the behavior of the application is highly data dependent, we measured the performance of each P_{ij} using input streams composed of 1000 frames for four different input scenarios, each causing a different workload: 1) a stream capturing a scene without any person; 2) a stream where a subset of the frames contains people; 3) a stream where a person appears in each frame; and 4) a stream where two people appear in each frame.

The bar diagrams of Fig. 2 report the average values of the execution times of the alternative partitions P_{ij} obtained by running the embedded software on the three processors of Table III. Specifically, the four parts of each bar corresponds to the average time across 1000 experimental runs that is necessary to complete the execution of the tasks belonging to P_{ij} on one video frame for one of the four possible input scenarios. Notice that the software implementation and the choice of the compiler were not optimized for any processor. This explains the similar ratios between two P_{ij} across the various processors. Clearly, most of the time is spent executing A_1, A_2 and A_3. Fig. 2(d) summarizes the memory taken by each P_{ij} when running on the ARM 9 processor. The results for the other processors are similar because the code footprint has small variations among architectures. The peak usage of the data memory strongly depends on the given scenario.

In this application, the payload at the output of a sequence of tasks decreases when we move down in the chain (i.e. for larger i). For the same i, the payload increases with the complexity of the scene (e.g. for large E).



(a) CORE 2 DUO2 (b) POWERPC (c) ARM 9 (d) Memory Usage

Fig. 2. Embedded software profiling results: average values of frame execution time (a,b,c) and peak memory usage (d).

IV. MAPPING ON EMBEDDED PLATFORMS

Based on the embedded software profiling, we derived a set of alternative embedded platforms by combining off-the-shelf components [14], [15], [16] and standard network technologies. Each platform can support the mapping of some subset of the task partitions P_{ij}. We chose the cheapest architecture that can execute the longest sequence of tasks within the frame period F. We assumed that each video-node is equipped with a simple PIR sensor that allows to dynamically save computational effort whenever it does not detect any person presence/movement. In general, the deployment of a camera-network application can be dimensioned based on a worst-case design (more common in security or control applications) or an average-case design. We capture this difference by introducing a parameter E that denotes the probability of detecting a presence/movement. We set $E = 1$ for a system that must process/transmit each frame of the video stream (worst case). A smaller E indicates that only a fraction of the frames are analyzed/transmitted. Each mapping gives specific communication requirements that the network connecting the video-nodes must satisfy. For the network implementation we considered two field buses that are widely used in building automation and well suited for this application class [17]: ARCnet (with EIA-485 as physical layer) [18] and Ethernet [19]. 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 offers a larger communication bandwidth, which makes it more attractive for video streaming applications.

Raw Transmission Central Computation (RTCC). This platform combines a video sensor and a low-cost micro-controller. Each pixel of each frame is forwarded directly from the sensor to the communication channel. Only tasks A_{VS} and A_{MD} are mapped on this platform while the other tasks are executed by an intermediate processor (e.g. CH).

Embedded Computation. The micro-controller implements the communication stack and oversees the PIR and the camera. The choice of the specific device is driven by the choice of the network technology because the micro-controller must sustain the necessary communication bandwidth.

Communication. The transmission rate depends on the occurrence of a movement event. The average payload is $B(A_{MD}) \simeq 2 \cdot E$ Mbps. ARCnet can be used only for small values of E.

Memory. A small frame buffer is required to provide flexibility in implementing the communication. Its size is larger in the case of ARCnet and for high values of E.

Partially Distributed Computation 1 (PDC1). Tasks A_{VS}, A_{MD} and A_1 can be mapped onto this platform. The gateway processor executes the remaining tasks.

Embedded Computation. According to the software profiling, an ARM 9 at 150 MHz executes A_1 in less than a frame period of 40 ms. Hence, the NXP LPC3180 with a 208 MHz ARM 9 core is sufficient for the three mapped tasks.

Communication. The considerations made for RTCC hold for PDC1 because they produce the same payload.

Memory. One MB is needed to execute A_1 and a second MB to execute the other two tasks and support the frame buffer and a small footprint embedded operating system.

Partially Distributed Computation 2 (PDC2). Tasks A_{VS}, A_{MD}, A_1 and A_2 can be mapped on PDC2.

Embedded Computation. To execute A_1 and A_2 in a frame period requires an ARM 9 at 250MHz. The Atmel AT91SAM combines the ARM 9 core with an useful camera interface.

Communication. The average payload produced by A_2 depends on the scene and is $B(A_2) \simeq 365 \cdot E$ Kbps.

Memory. Tasks A_2 consumes a relatively large amount ($\sim$500 KB) of memory, thus requiring a 64 Mb SDRAM.

Completely Distributed Computation (CDC). The computational tasks are all mapped on the video-node that must be capable of processing the scene and counting people. Only the last task A_{OC} runs on the central gateway processor that receives just a few bytes per second from each video-node.

Embedded Computation. To execute the whole video processing in real-time we need a ARM 9 clocked at least at 370 MHz. The Freescale iMX27 is the candidate architecture running at 400 MHz and featuring also an integrated camera interface and a hardware H.264 engine.

Communication. Since only few bytes per second are sent to the gateway, either Ethernet or ARCnet can be used.

Memory. To support the execution of all tasks we need a faster memory, i.e. a 200 MHz SDRAM.

Cluster Head (CH). This node has the capability to process the images captured by its video-sensor plus some streams coming from other cameras. The higher processing requirements are addressed by a 600 MHz PowerPC750.

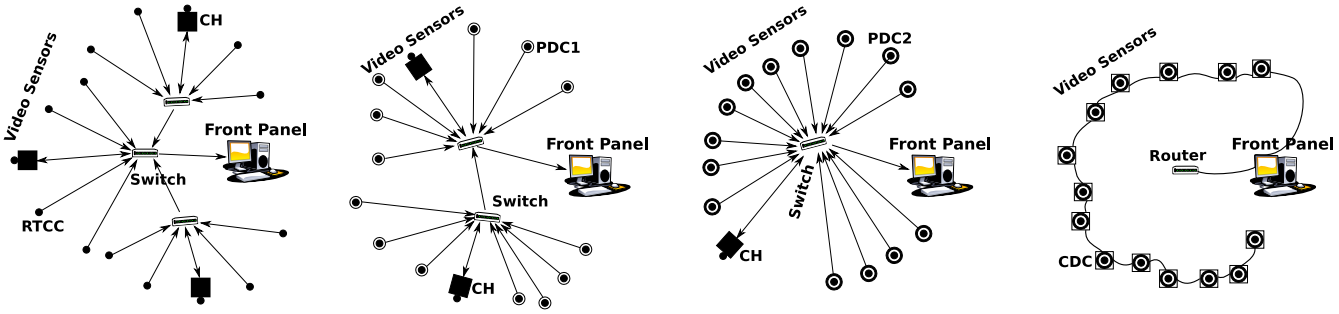
Embedded Computation. The number of streams that can be processed depends on the number of tasks that are executed on the other nodes.

Communication. There are two (possibly distinct) interfaces: one to gather streams from the video-nodes and one to communicate with the central gateway/front panel.

Memory. This node uses a 400 MHz DDR memory.

Embedded Platform	Mapped Actors	Micro-controller	Memory		PHY	PHY Chip	PIR ($)	Total ($)
			NOR FLASH (Mb)	RAM (Mb)				
ARCnet enabled								
RTCC	A_{VS},A_{MD}	LPC2131 @ 4 $	integrated	SRAM 2 Mb @ 2 $	EIA-485	ADM1485 @ 2$	2$	10
PDC1	A_{VS},A_{MD}, A_1	LPC3180 @ 8.5 $	16 @ 1.5 $	SDRAM 64Mb @ 2.5 $				16.5
PDC2	A_{VS},A_{MD}, A_1, A_2	AT91SAM @ 10 $	32 @ 2 $	SDRAM 64 Mb (@150 MHz) @ 2.5 $				18.5
CDC	A_{VS},A_{MD}, A_1, A_2, A_3, A_4, A_5	i.MX27 @ 15 $	32 @ 2 $	SDRAM 64 Mb (@200 MHz) @ 3 $				24
Ethernet enabled								
RTCC	A_{VS},A_{MD}	LPC2364 @ 5 $	integrated	SRAM 2 Mb @ 2 $	Eth 10/100	PC82562EP @ 3$	2$	12
PDC1	A_{VS},A_{MD}, A_1	LPC3180 @ 8.5 $	16 @ 1.5 $	SDRAM 64Mb @ 2.5 $				17.5
PDC2	A_{VS},A_{MD}, A_1, A_2	AT91SAM @ 10 $	32 @ 2 $	SDRAM 64 Mb (@150 MHz) @ 2.5 $				19.5
CDC	A_{VS},A_{MD}, A_1, A_2, A_3, A_4, A_5	i.MX27 @ 15 $	32 @ 2$	SDRAM 64 Mb (@200 MHz) @ 3 $				25
CH	see Sec. V	PowerPC750 @ 31 $	32 @ 2 $	DDR 256 Mb (@400 MHz) @ 6 $	NIC	10 $	2 $	51

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



(a) 12 RTCC and 3 CH nodes interconnected with an Ethernet bus

(b) 13 PDC1 and 2 CH nodes interconnected with an Ethernet bus

(c) 14 PDC2 and 1 CH nodes interconnected with an Ethernet bus

(d) 15 CDC nodes interconnected with an ARCnet bus

Fig. 3. Alternative application mappings for the case of 15 video-nodes and one front panel.

V. Implementation Cost Model

Table III summarizes the costs of the embedded platforms presented in Section IV. These platforms can be combined and interconnected in different ways to support the target application. We considered four modular and scalable combinations: CDC, RTCC and CH, PDC1 and CH, PDC2 and CH. Fig. 3 shows examples of alternative application mappings for a network of fifteen cameras and one front panel, which executes the A_{OC} task. The computing power of the video-node platform grows as we move from Fig. 3(a) to Fig. 3(d).

A CH node can process S additional streams beside its own. The value of S can be estimated by profiling the execution time $T(P_{ij})$ of a task combination P_{ij} on CH as described in Section III. Specifically, $T(P_{15}) \simeq 5.8$ ms, $T(P_{25}) \simeq 4.1$ ms and $T(P_{35}) \simeq 1.8$ ms. The overhead due to context-switch, communication and for A_{VS} and A_{MD} can be conservatively estimated as $H = 10$ ms. Hence, $S(i,j)$ is given by:

$$S(i,j) = min\left(\left\lfloor \frac{W_c}{B(A_j)\cdot E}\right\rfloor, \left\lfloor \frac{F-H-T(P_{15})\cdot E}{T(P_{ij})\cdot E}\right\rfloor\right)$$

where F is the frame period, W_c is the bandwidth of the communication technology (slow ARCnet with W_0=78.5 Kbps; fast ARCnet with W_1=2.5 Mbps; Ethernet: W_2=100 Mbps). Next, we detail the costs of the four combinations of embedded platforms and the analytical model for the network costs. To highlight the differences among the solutions, our implementation cost model omits the costs of the image sensor and the installation of the video-nodes.

CDC. This platform supports a fully-distributed computation. The cost of the hardware is

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

where C_G is the cost of the gateway processor, N is the number of cameras, and C_{CDC_c} is the cost of a CDC video-node. The index c distinguishes a video-node supporting ARCnet ($c = 0, 1$) from one supporting Ethernet ($c = 2$).

RTCC and CH, CDC1 and CH, CDC2 and CH. The computation costs of these platforms are captured by:

$$C_{comp}^{z} = C_G + N \cdot C_{z,c} + (C_{CH} - C_{z,c}) \left\lceil \frac{N}{1+S(z,5)} \right\rceil \quad (2)$$

where $z = 1, 2, 3$ indicates the RTCC, CDC1 and CDC2 video-node, respectively and $C_{z,c}$ the associated cost. C_{CH} is the cost of the CH architecture.

Network Cost Models. Cabling accounts for a significant fraction of the cost of a wired-networked embedded system. Many factors influence the cabling cost, including 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: slow ARCnet, fast ARCnet, Ethernet, and two hierarchical heterogeneous networks (Ethernet over slow ARCnet and Ethernet over fast ARCnet).

We assume that N cameras are distributed uniformly in a region of area A with density $\rho = \frac{N}{A}$. Then, the distance between two adjacent sensors is approximately $l = 2\sqrt{\frac{1}{\pi\cdot\rho}}$. For simplicity, we place the gateway at the center of the region.

ARCnet. A network based on ARCnet technology connects all video-nodes with daisy-chain buses. The cabling cost is proportional to the length of the chain $C_{cab}^{A} = (N-1)\cdot l \cdot C_{ARC}$. Since both slow and fast ARCnet networks use the same wiring medium and connection topology, they have the same cabling cost. However, on a slow ARCnet chain there can be at most 32 stations, whereas on a fast ARCnet chain this number is reduced to 8. Routers are used to connect multiple chains. In summary the ARCnet equipment costs are $C_{dev}^{A78} = \left\lceil \frac{N}{32} \right\rceil \cdot C_{AR}$ and $C_{dev}^{A25} = \left\lceil \frac{N}{8} \right\rceil \cdot C_{AR}$.

Second Tier Network	First Tier Network: slow ARCnet (@78$Kbps$)	First Tier Network: fast ARCnet (@2.5$Mbps$)
fast ARCnet (@2.5$Mbps$)	$C_{cab}^{AII} = (M-1)\cdot R_L \cdot C_{ARC} + C_{cab}^{A}$ $C_{dev}^{A25II} = \lceil \frac{M}{8} \rceil \cdot C_{AR} + C_{dev}^{A78}$ $R_L = 2\sqrt{\frac{N}{M\pi\rho}}$ $M = \lceil \frac{N}{32} \rceil$	$C_{cab}^{AII} = (M-1)\cdot R_H \cdot C_{ARC} + C_{cab}^{A}$ $C_{dev}^{A25II} = \lceil \frac{M}{8} \rceil \cdot C_{AR} + C_{dev}^{A25}$ $R_H = 2\sqrt{\frac{N}{M\pi\rho}}$ $M = \lceil \frac{N}{8} \rceil$
Ethernet	$C_{cab}^{Eth\_arc78} = [L_{EA}(J)\cdot \lfloor \frac{M}{J} \rfloor + L_{EA}(M \bmod J) + \frac{1}{2}\cdot\sqrt{\frac{N}{\pi\cdot\rho}}\cdot\lceil\frac{M}{J}\rceil]\cdot C_{ETH} + C_{cab}^{A}$ $C_{dev}^{ETH\_arc78} = \lceil \frac{M}{J} \rceil \cdot C_{ES} + C_{dev}^{A78}$ $L_{EA}(i) = 2\cdot R_L \cdot d_i$ $M = \lceil \frac{N}{32} \rceil$	$C_{cab}^{Eth\_arc25} = [L_{EA}(J)\cdot \lfloor \frac{M}{J} \rfloor + L_{EA}(M \bmod J) + \frac{1}{2}\cdot\sqrt{\frac{N}{\pi\cdot\rho}}\cdot\lceil\frac{M}{J}\rceil]\cdot C_{ETH} + C_{cab}^{A}$ $C_{dev}^{Eth\_arc25} = \lceil \frac{M}{J} \rceil \cdot C_{ES} + C_{dev}^{A25}$ $L_{EA}(i) = 2\cdot R_H \cdot d_i$ $M = \lceil \frac{N}{8} \rceil$

Table IV: Hierarchical heterogeneous network model.

parameter	definition	value
F	frame period	40 ms
A	area of the building	2100 m^2
C_{ARC}	$[\$/m]$: cost per meter of an ARCnet cable	\$3
C_{ETH}	$[\$/m]$: cost per meter of an Ethernet cable	\$4.5
J	number of port in the Ethernet switch	8
C_G	cost of the gateway/front panel	\$300
C_{ES}	Ethernet switch cost	\$150^a
C_{AR}	ARCnet router cost	\$460^a
	^{a}) plus \$ 100 of installation cost	

Table V: Cost model parameters.

Two-Tier Switched Ethernet. In the first tier signals from J sensors are merged together by the switch. The second tier connects all the switches to the gateway with one additional switch. Assuming that each switch is installed at the center of its group of sensors, the Ethernet cable length to connect i sensors to a switch is approximated by $L_E(i) = 2\cdot l \cdot d_i$ where d_i is the i-th element of D. In case of an 8-port Ethernet switch $D = (0, 1, 2, 5, 8, 10, 13, 16)$. The total cable cost is:

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

The first two terms account for the wiring needed by N cameras. The third term models the cable connecting switches and gateway. C_{ETH} captures installation and cable cost per meter. The switches contribute $C_{dev}^{ETH} = \lceil \frac{N}{J} \rceil \cdot C_{ES}$.

Hierarchical heterogeneous networks. In these networks the first tier (closer to cameras) and the second tier are implemented with different technologies. The cabling and component cost of the first-tier networks are the same as above. The second-tier network is commonly realized with a wide bandwidth field bus (e.g. Ethernet or fast ARCnet). We considered the combinations of Table IV.

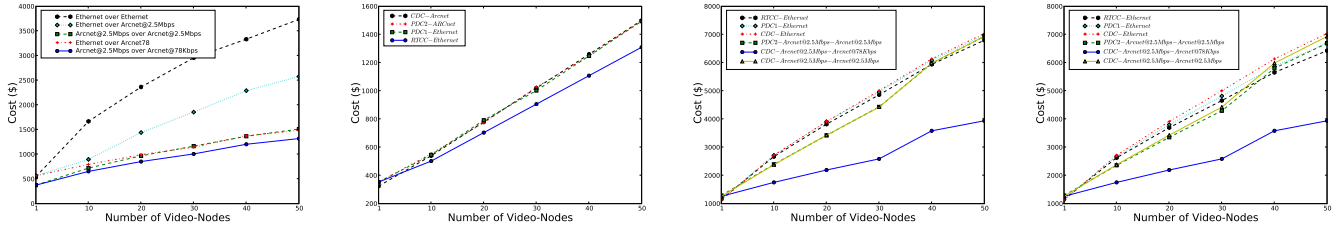
VI. Experimental Results

Using the reference values for the cost model reported in Table III and V we derived model-based estimates of the implementation costs for the various implementation technologies described above as function of the number of video-nodes in the building. The cabling cost is shown in Fig. 4(a). The Ethernet solution turns out to be the expensive compared to any other solutions. The reason is twofold: (1) Ethernet uses a star topology that generally requires more cabling than the ARCnet daisy chain buses and (2) the installation cost of an Ethernet cable is higher than the ARCnet counterpart. While a pure ARCnet network seems to be the most cost effective, the curves for the hierarchical heterogeneous networks show that combining the two technologies may offer a good compromise between bandwidth and price. Fig. 4(b) shows the computation cost of various platforms based on Eq. 1 and 2 for the case $E = 1$, i.e. the worst-case design where each frame of each stream must be processed. These curves indicate that a solution based on RTCC and CH nodes is $\sim 20\%$ cheaper than all the others. Moreover, a smaller value of E increases this cost saving. Finally, Fig. 4(c) and 4(d) show the total implementation costs for $E = 1$ and $E = 0.1$ respectively, including the costs of communication infrastructure and computation nodes. These figures help us to understand the main design trade-offs: *the more computation is performed locally on the video-node, the smaller bandwidth constraints are imposed on the network.* Hence, more investments in the video-node enable big savings in the network design. The estimations based on the analytical model with the parameters of Table V show that the cabling costs play a critical role to tilt the balance in favor of a distributed implementation. Specifically, the CDC platform with the slow ARCnet network is the most convenient solution regardless of the number of sensors in the system.

The analytical model gives a general estimation of the implementation costs. In practice, these can be affected by the characteristics of a particular deployment such as building layout and camera positions. We performed a second experiment by deriving detailed network topology implementations for two real buildings with very different layouts (Fig. 5(a) and 5(c)). Fig. 5(b) and 5(d) report the total deployment costs for the case $E = 0.1$. Comparing these results with those based on the analytical model, we notice that the CDC platform with the slow ARCnet network is still the best solution in terms of costs regardless of the building. Also, the analytical model predicts with a good approximation the cost of Building B, which has a regular structure that better matches the modeling hypothesis, while it underestimates those of Building A. Finally, the network cost, which is a significant component of the total cost, is clearly dependent on the building layout.

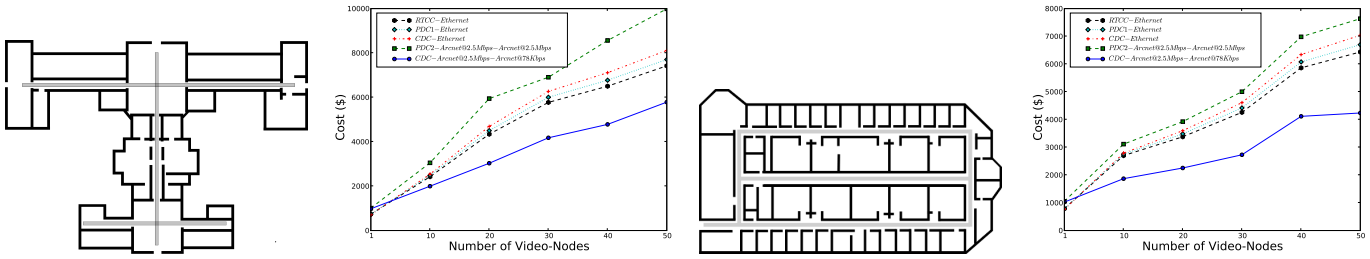
VII. Concluding Remarks

In the building occupancy estimation the information content produced by each camera can be summarized in a few bytes per second. Thanks to this huge data reduction and to the substantial independence of the task computations across the network, it is convenient to move the bulk of the computation as close as possible to the cameras. This choice is also supported by trends in embedded computing that steadily offer new low-cost, low-power high-performance embedded processors. However, other camera-network applications may require that more feature-rich data are transmitted across the network. For example, our camera network can be easily adapted to



(a) cabling costs (b) computation costs with E=1 (c) total costs with E=1 (d) total costs with E=0.1

Fig. 4. Implementation cost estimates per number of video-nodes based on the analytical model.



(a) Building A layout (b) Building A total cost (c) Building B layout (d) Building B total cost

Fig. 5. Layout of two real buildings and corresponding total deployment costs with $E = 0.1$.

implement an intrusion detection/control system. In this case it is important not only to have an occupancy estimation, but also to view/record frames/streams from those cameras that detect movements. This can be seen as an extension of the previous application where both occupancy estimation and video streaming are now simultaneously present. While the application is essentially made of the same embedded software tasks, the bandwidth constraints are quite different. This has major consequences on possible design implementations.

First, transmitting a raw video stream on a low-bandwidth network is clearly unfeasible. Second, to compress and transmit multiple video streams on a ARCnet network would require image compression techniques (e.g., H.264). Specifically, the low resolution video produced by our 3D video sensor can be encoded in a $\sim$30 Kbps stream. Here we can distinguish two cases: a) In an average-case design with a small probability E of movement events, the compressed video can be transmitted on a slow ARCnet network and the video-node and communication costs are similar to those relative to CDC over ARCnet plotted in Fig. 5(b) and 5(d). b) In a worst-case design scenario (common in security or control applications) or when $E \sim 1$, the bandwidth produced by the camera network ($\sim$ 30 Kbps per sensor) rules out the option of using the slow ARCnet network, which supports only 78 Kbps. Instead, the cameras must be connected with a fast ARCnet network that can sustain a large number of compressed streams. However, this choice turns out to be more expensive than using Ethernet because it would present a similar cost as the cost of PDC2 shown in Fig. 5(b) and 5(d). Also, naturally, to compress and decode a stream requires dedicated hardware and software components, thus increasing the global implementation cost.

Finally, the wide bandwidth offered by Ethernet can be used to transmit raw videos. This choice enables the possible cost-savings related to the aggregation of the computation on CH nodes. However, beyond a given number of video-nodes the total bandwidth load on the network becomes too large for a standard 10/100 Ethernet field-bus.

ACKNOWLEDGMENTS

This research is sponsored in part by the National Science Foundation (under Award #: 0644202).

REFERENCES

[1] W. Wolf, B. Ozer, and T. Lv, "Smart cameras as embedded systems," *Computer*, vol. 35, no. 9, pp. 48–53, 2002.
[2] M. Bramberger *et al.*, "Distributed embedded smart cameras for surveillance applications," *IEEE Computer*, vol. 39, no. 2, pp. 68–75, 2006.
[3] B. Rinner and W. Wolf, "An introduction to distributed smart cameras," *Proc. of the IEEE*, vol. 96, no. 10, pp. 1565–1575, Oct. 2008.
[4] R. Tomastik, Y. Lin, and A. Banaszuk, "Video-Based Estimation of Building Occupancy During Emergency Egress," in *Proc. of the American Control Conference*, 2008, pp. 894–901.
[5] Axis. [Online]. Available: http://www.axis.com
[6] Biodata. [Online]. Available: http://www.videoturnstile.com
[7] Honeywell. [Online]. Available: www.honeywell.com
[8] Eurotech. [Online]. Available: http://www.eurotech.com
[9] Mate Webpage. [Online]. Available: http://www.mate.co.il
[10] A. Bevilacqua, L. Di Stefano, and P. Azzari, "People Tracking Using a Time-of-Flight Depth Sensor," in *Proc. of the IEEE Intl. Conf. on Video and Signal Based Surveillance*, 2006.
[11] D. Beymer, "Person counting using stereo," in *Proc. of the Workshop on Human Motion (HUMO'00)*, 2000, p. 127.
[12] K. Terada *et al.*, "A counting method of the number of passing people using a stereocamera," in *IEEE Proc. of Industrial Electronics Conf.*, vol. 3, 1999, pp. 1318–1323.
[13] L. Marcenaro *et al.*, "Distributed architectures and logical-task decomposition in multimedia surveillance systems," *Proc. of the IEEE*, vol. 89, no. 10, pp. 1419–1440, 2001.
[14] Avnet. [Online]. Available: http://www.avnet.com/
[15] Power User Inc. [Online]. Available: http://www.poweruseronline.com
[16] Grid Connect. [Online]. Available: http://www.industrialethernet.com
[17] W. Kastner *et al.*, "Communication Systems for Building Automation and Control," *Proc. of the IEEE*, vol. 93, no. 6, pp. 1178–1203, 2005.
[18] ARCnet Trade Association. [Online]. Available: http://www.arcnet.com
[19] Ethernet Specification. [Online]. Available: http://standards.ieee.org/getieee802/802.3.html