Increasing Service Availability in Intelligent Video Surveillance Systems by Fault Detection and Dynamic Reconfiguration

A. Doblander, A. Maier, B. Rinner
{doblander, maier, rinner}@iti.tugraz.at
Institute for Technical Informatics
Graz University of Technology
Graz, AUSTRIA

H. Schwabach
helmut.schwabach@arcs.ac.at
ARC seibersdorf research
Seibersdorf, AUSTRIA

Abstract – In this paper, we present an approach for increasing service availability in intelligent video surveillance systems (IVS). A typical IVS system consists of various intelligent video sensors that combine image sensing with video analysis and network streaming. System monitoring and fault diagnosis followed by appropriate system reconfiguration mitigate effects of faults and therefore enhance the system's fault tolerance. The applied monitoring and diagnosis unit (MDU) allows the detection of both node- and system-level faults. Lacking redundant hardware such reconfigurations are established by graceful degradation of the overall application. Multi-objective optimization is used to compute a new degraded system configuration by trading off quality of service (QoS), energy consumption, and service availability. We demonstrate our approach by typical scenarios in an IVS-system that necessitates reconfiguration.

Kurzfassung – In dieser Arbeit wird ein Ansatz zur Verbesserung der Dienstverfügbarkeit in intelligenten Videoüberwachungssystemen (IVS) präsentiert. Ein IVS-System besteht typischerweise aus mehreren intelligenten Videokameras die Szenen aufnehmen und diese Daten analysieren und über ein Netzwerk versenden. Ein Monitoring mit daran gekoppelter Fehlerdiagnose und entsprechender Rekonfiguration verbessert die Fehlertoleranz des gesamten Systems. Die verwendete Monitoring und Diagnose Einheit (MDU) ermöglicht das Erkennen von sowohl System- als auch Knoten-bezogener Fehlfunktionen. Bei nicht redundanter Hardware wird die Rekonfiguration durch angemessene Verringerung der Gesamtqualität im System erreicht. Die Verringerung wird im Detail durch optimierten Trade-off von Quality of Service (QoS), Energieverbrauch und Dienstverfügbarkeit bestimmt. Der Ansatz wird anhand eines typischen IVS-Szenarios das Reconfiguration nötig macht näher verdeutlicht.

Keywords: Intelligent video surveillance; distributed embedded real-time system; dynamic reconfiguration; energy-awareness; quality-of-service adaptation; fault tolerance, graceful degradation.

I. INTRODUCTION

A typical intelligent video surveillance (IVS) system consists of various intelligent video sensors that combine image sensing with video analysis and network streaming. The design of these processing units allows to yield various parameters of a captured scene and to compress a live video-stream simultaneously. It is, therefore, a distributed system of collaborating intelligent cameras. Typically, the system nodes (i.e. intelligent cameras) are implemented as embedded multi-processor systems operating autonomously.

Typical video analysis algorithms are our target application for traffic surveillance including motion detection, MPEG-4 encoding, tracking of objects, detection of stationary vehicles as well as the calculation of traffic statistics (such as average speed, number of cars etc.). Obviously, quality of service (QoS) is a major concern in video surveillance. Thus, IVS systems typically contain dedicated QoS-management mechanisms. Furthermore, QoS is closely related to power consumption making power-awareness an important design aspect as well. In recent work [1] we therefore take advantage of QoS-triggered dynamic power management.

Additionally, maintaining a high degree of service availability is also an important design goal for autonomous operation of an embedded distributed system [2]. This work focuses on improving service availability in IVS systems by enhancing its fault tolerance. System monitoring and fault diagnosis followed by appropriate system reconfiguration mitigate effects of faults. Lacking redundant hardware such reconfigurations are established by graceful degradation of the overall application. Multi-objective optimization is used to compute a new (degraded) system configuration by trading off QoS, energy consumption, and service availability.

II. RELATED WORK

Typical QoS-parameters in video surveillance are video data quality and its distortions in network transmission (jitter). Further parameters include quality metrics such as image size, data rate or blockiness or the number of frames per second (fps). However, parameters like the availability of the service are also taken into account as QoS-aspect. In case of low energy in parts of the system, the QoS gets seriously affected and degraded. Thus, there is a close link of these three parameters in IVS systems. In literature, there is an emphasis on investigating the trade-off between energy and QoS.

In [3] for instance, the authors investigate the trade-off between image quality and power consumption in wireless video surveillance networks. The work mainly focuses on the evaluation of sophisticated image compression techniques. However, existing implementations lack of comprehensive handling of these three correlating parameters.

Fault tolerance in embedded real-time systems is often achieved by checkpointing mechanisms. In [4], an adaptive checkpointing algorithm is proposed that also minimizes energy consumption. That is, a schedule is derived that includes the checkpointing task in a way that dynamic voltage scaling is most effective. In our system, however, we do not have the (non-volatile) memory capacity and the computational resources for a checkpointing approach.

In general purpose distributed computing it is common to use redundant hardware and employ load sharing techniques to increase fault tolerance (see, e.g., [5]). Given the tight cost constraints in the embedded market, however, we cannot afford hardware redundancy but depend on graceful degradation.

Similarly to [6] we also distinguish between system-level and application-level fault tolerance techniques. In this work the component faults, i.e., processor or sensor failures, are handled by system-level mechanisms. Whereas, inconsistent observations of multiple cameras are addressed by the application logic.

There are also several approaches for integrating fault tolerance techniques into the middleware layer of distributed real-time and embedded systems. The goal is to shift the trade-off between real-time execution and fault tolerance from design time to runtime to support the application developer [7]. In this work we have not currently spent much attention on that issue but we will focus on it in future work.

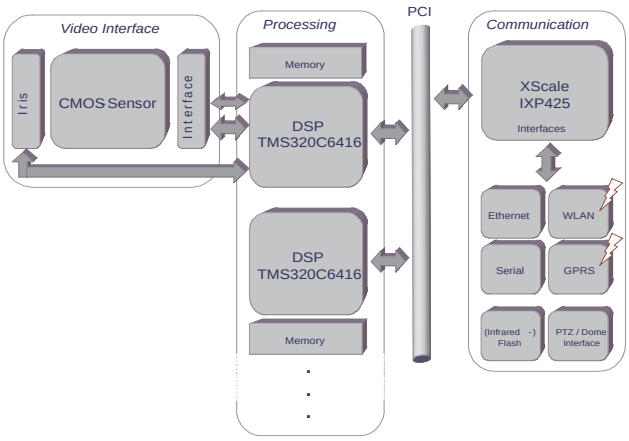

Fig. 1. Hardware Architecture of a Smart Camera Node of a Distributed IVS system.

III. SYSTEM DESCRIPTION

The considered video surveillance system is organized as a number of distributed *smart cameras* [8]. Each node is an embedded multi-processor platform equipped with a CMOS image sensor for video acquisition. A network processor (Intel XScale) and several DSPs (Texas Instruments TMS320C64x) provide the necessary communication capabilities and computing power for video analysis algorithms. Currently, the prototype node [9] is realized comprising two DSPs and one network processor that serves as the managing unit and connects the camera to an Ethernet network. All three processors are connected by a PCI bus. A VGA image sensor is directly connected to one DSP. Alternative network media such as GSM/GPRS or WLAN are also possible. For an overview of the node hardware architecture refer to Fig. 1.

Due to the heterogeneous processor hardware the software architecture comprises two major parts. The network processor hosts the so called *SmartCam-Framework* (SC-FW) which is run on top of Linux. As a counterpart on the DSPs the *DSP-Framework* (DSP-FW) is run on top of the DSP/BIOS operating system. A block diagram of the software architecture is depicted in Fig. 2.

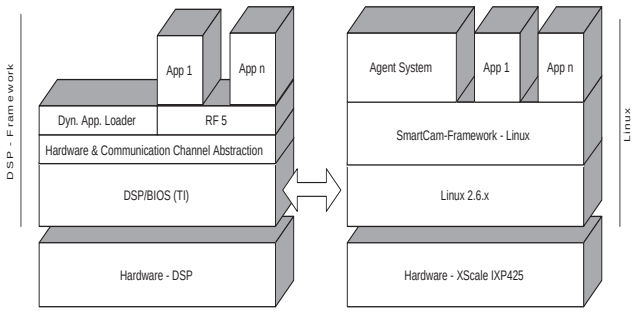

Fig. 2. Software Architecture of the Prototype Node.

A key functionality in the software framework is the support for task migration. The *dynamic loader* (DL) provides dynamic linking for the DSPs and, therefore, allows for software reconfiguration during runtime. Tasks can be migrated to another DSP on the same node or to a DSP on a remote node.

To assure eligible migrations there is a *resource manager* (RM) that keeps track of important system resources—CPU load, memory usage and DMA channel allocations for each processor, as well as PCI bus and network utilization for the overall node. Using this information it can be determined if there are sufficient resources on the target host for a planned migration.

The actual applications on the smart camera are video analysis and compression algorithms. Currently, four algorithms are considered:

- *Motion detection* to detect motion, camera blackouts and whiteouts.
- *MPEG-4* encoder for video compression
- *Stationary vehicle detection*, which can also be used for detecting lost cargo.
- *Traffic statistics* for computing average traffic speed, driving lane utilization and more
- *Vehicle Tracking* (VT) e.g. to track hazardous-cargo vehicles along tunnels.

Unfortunately, the above algorithms are very demanding with respect to computing resources and so it is hardly possible that all run simultaneously. However, not all algorithms are equally important at all times. So it is possible to reduce the quality-of-service (QoS) of less important components to permit others to be run too. Different quality levels are distinguished in terms of frame rate and image size. Each algorithm has to support three different QoS-levels Q_i:

- Full quality (Q_1)
- Reduced quality (Q_2)
- Minimum quality (Q_3)

Algorithms provide a dedicated interface for adjusting QoS-level settings. A QoS-level switch can be initiated by an algorithm itself or by the system software in reaction to special events.

IV. MONITORING AND DIAGNOSIS

A key requirement for dynamically reacting to faults and failures is to detect abnormal behavior and isolate affected system components. It is important for the reconfiguration process to know which resources are not available after a fault has occurred. It is the responsibility of a special monitoring and diagnosis unit (MDU) to indicate faulty system behavior and present a diagnosis to the configuration manager as seen in Fig. 3:

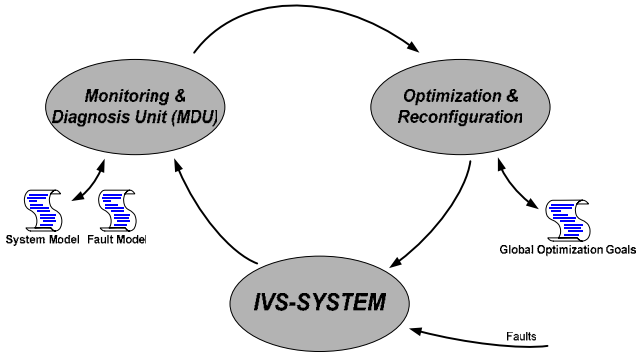

Fig. 3. Monitoring, Diagnosis and Reconfiguration.

The major concern of this work is on establishing eligible system configurations in case of faults. Monitoring and diagnosis techniques applied to yield necessary diagnostic information are, therefore, only briefly described.

The IVS system as described in Sec. III can be viewed from two perspectives. First, the *system view* considers the overall distributed application. That is, all nodes (hardware and system software) including all algorithms running on these nodes. Second, the *node view* considers only a single node.

According to these two different views the monitoring and diagnosis process comprises also a system-level part and a node-level part. For local faults only the MDU of the corresponding node is involved in the monitoring and diagnosis procedure. Node-level faults that are currently diagnosable are

- Crash-faults of DSPs
- Crash-fault of the CMOS-sensor
- Crash-faults of algorithms
- (memory leaks).

More interesting, however, are the system-level faults. Detection of system-level problems involves multiple nodes sharing monitoring information and exploiting application specific knowledge. Faults of this category that are currently diagnosable are

- Value-faults of instances of the stationary vehicle detection algorithm
- Value-faults of instances of the traffic statistics algorithm

To detect and diagnose such erroneous algorithm behavior a majority decision is employed by comparing results of three neighboring nodes. Because in typical traffic surveillance applications it can be expected that cameras within a dedicated geographical area observe very similar events. Assuming that all camera nodes are arranged in regular intervals alongside a freeway (or tunnel) observations of neighboring cameras are equal but appear at different times. The distributed diagnosis is performed by a simple communication protocol based on [10] and [11].

V. OPTIMIZATION AND TRADE-OFFS

It is the optimizer's task to compute a system configuration by multi-objective optimization. By adding "maximum availability" to "minimum energy consumption" and "maximum QoS" we get a total of three different optimization criteria. As these are conflicting criteria the resulting system configuration will always be subject to a trade-off.

In general, determining a configuration is a mapping of intended functionality onto remaining system resources. For setting a specific system configuration the optimizer uses the framework's capabilities for dynamic software reconfiguration as previously described.

Depending on the actual system’s requirements, the goal of optimization may vary in between the three objectives QoS, energy and availability. In a typical surveillance scenario for instance, the system may execute in full quality (Q_1) with no respect to energy efficient execution but to maximum service availability and proper function of the overall-system. This quality level is usually used for human supervision whenever detailed scene information is required like in critical situations such as accidents.

In order to maximize energy savings, the quality level may get degraded to Q_2. This quality level still delivers proper input for sufficient function of the traffic analysis algorithms but in a more energy-aware way. Nevertheless, if a system-level fault appears, the optimizer needs to force a proper reconfiguration due to the sort of malfunction. In case of an incorrect recognition of stationary vehicles the algorithm in the corresponding node needs to be restarted. In case of damage or not applicable resetting of a node due to hardware errors, the quality may need to be degraded to the minimum quality level Q_3 to fulfill the optimization goal of minimum availability.

VI. CASE STUDY

We demonstrate our approach by a typical IVS-setup. It consists of seven smart cameras that execute various tasks and are equipped along a one-way road in a tunnel as depicted in Fig. 4:

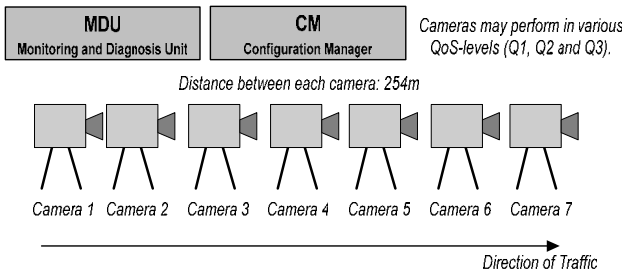

Fig.4. A Typical IVS-Setup along a Tunnel.

The following scenario demonstrates the objective of optimal QoS delivery. While all cameras perform in Q_2 QoS-level, camera #3 detects a stationary vehicle. Furthermore, the traffic statistic algorithms of the successive (in direction of traffic) cameras #4 to #7 report a sudden decrease of volume of traffic. These parameters indicate a critical situation in the controlled area of camera #3 and therefore force the MDU to generate an alert to the human interface. The configuration manager then immediately forces the corresponding device (camera #3) to increase its QoS to full quality (Q_1) in order to have optimal surveillance conditions for the human interface. However, cameras #4 to #7 can then reduce their quality level to minimum quality (Q_3) to save energy.

In the next scenario we focus on detecting a value-fault caused by an instance of the stationary vehicle detection algorithm. This results in a system-level fault detected by the MDU that needs to be maintained by the configuration manager. In this case, camera #5 reports a detected stationary vehicle, while all other cameras still report normal motion. In particular, the traffic statistics behind camera #5 (i.e., of camera #6 to #7) indicate normal volume of traffic in all surveillance areas.

However, the MDU recognizes this as possible incorrect configuration and forces the configuration manager to reset and restart the SVD algorithm on camera #5. Nevertheless, the event generates a message to the human interface to inform the supervisor.

Another scenario shows the behavior of the optimizing unit in order to reduce the overall energy consumption. In this case, the statistics manager reports a decreasing volume of traffic over a longer period of time while still executing all cameras in full quality level (Q_1). The MDU then forces the configuration manager to reduce the overall quality level to minimum quality (Q_3) and allows to maximize energy savings.

VII. CONCLUSION

In this ongoing work we investigate a special method for increasing service availability in intelligent video surveillance (IVS) systems. Therefore, monitoring and diagnosis capabilities are added to the system. In case of faults a multi-objective optimization process is used to determine a new degraded system configuration maintaining as much functionality as possible. Mechanisms developed in previous work are used for actual system reconfiguration during runtime. In a practical example we demonstrate the concept of the approach.

In further work we intend to extend our approach to include means for achieving bounded detection latency and bounded recovery time. Especially for safety-critical applications like traffic surveillance it is important to guarantee well defined time bounds also for a distributed implementation. We also intend to better integrate the fault tolerance mechanism into our software framework (i.e. middleware) to make them less dependent on a specific application and support application development.

VIII. REFERENCES

[1] A. Maier, B. Rinner, T. Trathnigg and H. Schwabach, "Combined Management of Power-and Quality of Service in Distributed Embedded Video Surveillance Systems", *First Int'l Workshop on Power-Aware Real-Time Computing*, September 2004, Pisa, Italy

[2] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing", *IEEE Computer*, vol. 36, nr. 1, January 2003, pp. 41-50

[3] C.F. Chiasserini and E. Magli, "Energy Consumption and Image Quality inWireless Video-Surveillance Networks", in *Proceedings of the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2002.

[4] Y. Zhang and K. Chakrabarty, "Energy-Aware Adaptive Checkpointing in Embedded Real-Time Systems", in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition* (DATE'03), 2003

[5] L. Lundberg, D. Häggander, K. Klonowska and C. Svahnberg, "Recovery Schemes for High Availability and High Performance Distributed Real-Time Computing", in *Proceedings of the International Parallel and Distributed Processing Symposium* (IPDPS'03), 2003

[6] J. Haines, V. Lakamraju, I. Koren and C. M. Krishna, "Application-Level Fault Tolerance as a Comlement to System-Level Fault Tolerance", *Journal of Supercomputing* (Kluwer), vol. 16, 2000, pp. 53-68

[7] T. Bracewell and Priya Narasimhan, "A Middleware for Dependable Distributed Real-Time Systems", in *Proceedings of the Joint Systems and Software Engineering Symposium*, 2003.

[8] W. Wolf, B. Ozer and T. Lv, "Smart Cameras as Embedded Systems", *IEEE Computer*, vol. 35, nr. 9, September 2002, pp. 48-53

[9] M. Bramberger, J. Brunner, B. Rinner and H. Schwabach, "Real-Time Video Analysis on an Embedded Smart Camera for Traffic Suveillance", in *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium* (RTAS'04), 2004

[10] J. G. Kuhl and S. M. Reddy, "Fault-Diagnosis in Fully Distributed Systems", in *Proceedings of the 11th IEEE International Symposium on Fault-tolerant Computing* (FTCS-11), pp. 100-105, 1981

[11] A. Subbiah and D. M. Blough, "Distributed Diagnosis in Dynamic Fault Environments", *IEEE Trans. on Parallel and Distributed Systems*, vol. 15, nr. 5, May 2004, pp. 453-467