

A real-time networked camera system

Citation for published version (APA):

Karatoy, H., & Technische Universiteit Eindhoven (TUE). Stan Ackermans Instituut. Software Technology (ST) (2012). *A real-time networked camera system: a scheduled distributed camera system reduces the latency*. [EngD Thesis]. Technische Universiteit Eindhoven.

Document status and date:

Published: 01/01/2012

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

A Real-Time Networked Camera System

Hilal Karatoy
March 2012



A Real-Time Networked Camera System

Hilal Karatoy

March 2012

A Real-Time Networked Camera System

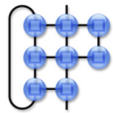
A scheduled distributed camera system reduces the latency

H. Karatoy

Eindhoven University of Technology

Stan Ackermans Institute/ Software Technology

Partners



System Architecture
and Networking Group



Eindhoven University of Technology

Steering Group

Prof. Antonio Liotta

Prof. J. Johan Lukkien

Date

March 2012

Contact Address Eindhoven University of Technology Department of Mathematics and Computer Science
HG 6.57, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
+31402474334

Published by Eindhoven University of Technology Stan Ackermans Institute

Printed by Eindhoven University of Technology *Universiteits Drukkerij*

ISBN 978-90-444-1102-7

Abstract Modern video surveillance systems tend to use several networked cameras to observe different parts of a global scene. This induces a large data flow, which may lead to network congestion when transporting images to the servers. SAN Group provided a system, which is composed of multiple cameras and a PC running the distributed video processing application in order to prevent network congestion, while it satisfies the timing constraints of the video processing application. This report describes the design and implementation of a distributed real-time system that deals with both the resource reservation for the distributed video processing application running on the cameras and the real-time scheduling for the tasks comprising the distributed video processing applications.

Keywords Distributed System, Video Processing, Real-time, Scheduling, Admission Control, Network

Preferred Reference Hilal Karatoy, **A Real Time Networked Camera System**
Eindhoven University of Technology, SAI Technical Report, March, 2012, ISBN 978-90-444-1102-7 (Eindverslagen Stan Ackermans Instituut; 2011/064).

A catalogue record is available from the Eindhoven University of Technology Library.

Partnership	This project was supported by Eindhoven University of Technology and SAN Group.
Disclaimer Endorsement	Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the Eindhoven University of Technology or SAN Group. The views and opinions of authors expressed herein do not necessarily state or reflect those of the Eindhoven University of Technology or SAN Group, and shall not be used for advertising or product endorsement purposes.
Disclaimer Liability	While every effort will be made to ensure that the information contained within this report is accurate and up to date, makes no warranty, representation or undertaking whether expressed or implied, nor does it assume any legal liability, whether direct or indirect, or responsibility for the accuracy, completeness, or usefulness of any information.
Trademarks	Product and company names mentioned herein may be trademarks and/or service marks of their respective owners. We use these names without any particular endorsement or with the intent to infringe the copyright of the respective owners.
Copyright	<p>Copyright © 2011. . All rights reserved.</p> <p>No part of the material protected by this copyright notice may be reproduced, modified, or redistributed in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the and SAN Group.</p>

Foreword

From January 1 to November 2011, Hilal has been a member of the SAN group working on the demonstration of a camera system as described in this report. I am convinced that it has been an interesting period for everyone involved. Getting a good understanding of the problem, finding relevant information and getting the equipment to work, they all proved to be challenging. I have come to admire her perseverance, her absolute commitment to achieve her goals which I think is a very strong asset. The result now lies in the report before us which clearly shows that she has moved forward during the project.

Prof. Dr. Johan Lukkien

(Project Manager)

Preface

This report is the result of the “A Real-time Networked Camera System” project which was carried out by the author as part of her Industrial Design and Development project of the two-year post-master level Software Technology program (also known as OOTI¹) of the Stan Ackermans Institute (SAI).

The target audience of this report is a technical audience with a basic notion of modern software design and an interest in distributed real-time systems with video processing.

Readers who are interested in the distributed real-time system with video processing application and the original system and the problems with this system context can read Chapters 1 to 3. Readers that are interested in the technical design and implementation or those who wish to continue this project should go through Chapters 4 to 7. Chapter 8 lists the conclusions and recommendations of this project, while Chapter 9 contains details regarding the project’s management.

H. Karatoy

¹ Officially known as 3TU School for Technological Design Stan Ackermans Institute/ Software Technology, in Dutch: Ontwerpers Opleiding Technische Informatica.

Acknowledgements

Many people contributed in completion of this project. I am highly grateful to all the people who encouraged me and supported me throughout the project.

First of all, I would like to thank to my university supervisor Prof. Johan Lukkien for his great guidance and support. I am thankful for his patience in my work through the nine months of my study. I very much appreciate that he has thought along with me as much as he could. I have learned a lot from him, not just about technical matters but also on how to be a better person; being humble, being patience, being organized and being kind. His comments and remarks have always been very valuable. It was honor for me to work with him.

I am also truly thankful to Prof. Antonio Liotta who is my second university supervisor and project owner. I thank him for the fact that he has proposed this project and provided me with the opportunity to study in Real-Time Systems. Thanks to him I met and worked with remarkable people. I also thank to him for his supports, trust and excellent Italian coffee.

During the project, my office mates and the people that I shared the same floor with helped me a lot. I would like to give their names and express their importance to me: Norbert Verhagen for his Camera Platform explanations and all remarks on my English, Richard Verhoeven for his technical conversations and help, Mike Holenderski for his contribution to this project and also to the μ C/OS-II community, Martijn van de Heuvel and Reinder Bril for their technical discussions and the publications they provided which I used in my report as a reference, and Ugur Keskin for being a colleague and friend and I thank Rudolf Mak for his grammatical help and patience.

At OOTI, I want to thank the people who taught me how to design a system, how to work with multi cultural environment and how to manage the issues in the stressful environment. These are the people that I want to thank in specific: Dr. Ad Aerts, Harold Weffers, Onno van Roosmalen, Maggy de Wert and dutch instructor Nelleke.

I want to thank my colleagues and first foreign friends, especially Chris, Yogesh and Firat and my friend Selcuk Sandikci. Thank you for all technical discussions and personal support.

Also I want to express my gratitude to Ex-OOTIs, especially to Roza Akkus who gave me supported me tremendously, Lusine Hakobyan, Jorge Crespo, Oanna Dragomir, and Sunder Rao, Bui Vinh. These people listened and supported me and gave useful remarks on the project management from the very first day of the project.

And last but not least I want to thank my family with all my heart, for raising me with faith and love.

H. Karatoy

October, 2011

Executive Summary

This report presents the results of a Real-time Networked Camera System, commissioned by the SAN² Group in TU/e.

Distributed Systems are motivated by two reasons, the first reason is the physical environment as a requirement and the second reason is to provide a better Quality of Service (QoS). This project describes the distributed system with a video processing application. The aim is to deal with the distributed system as one system thus minimizing delays while keeping the predictability in a real-time context. Time is the most crucial ingredient for the real-time systems in the sense that the tasks within the application should meet with the task deadline.

With respect to the distributed system we need to consider a couple of issues. The first one is to have a distributed system and a modular application that is mapped to multiple system nodes. The second issue is to schedule the modules collectively and the third is to propose a solution when shared resource(s) (such as the network) are required by several nodes at the same time.

In order to provide a distributed system, we connect 2 cameras with 1 PC via a network switch. Video processing has two parts; the first part consists of creating a frame, encoding the frame, and streaming it to the network and the second part deals with receiving the frame, decoding the frame, and displaying the frame. The first part is running on the cameras and the second part is running on the PC.

In order to give real-time behavior to the system, the system components should provide the real-time behavior. The camera is installed with the μ C/OS-II (Open Source Real-time Kernel). We investigated the Real-time Operating System and its installation on the PC.

In order to provide resource management to the shared resources, we designed and implemented Admission control which controls access to the required connection to the PC. We designed and implemented a component to delay the start of any of the cameras in order to synchronize the network utilization. We also designed an enforcement component to allow the tasks to run as much as they should and monitor the frames streamed to the network.

The results show that with the Admission Control, cameras only send as many frames as the network can transport. The given start delay to the system shows that overlap can be prevented, but we could not evaluate it because of the semi-tested/unreleased code which is provided by the camera providers. The source code we used is the test source code which was not mature.

² Software Architecture and Networking

TABLE OF CONTENT

FOREWORD	VII
PREFACE	VIII
ACKNOWLEDGEMENTS.....	IX
EXECUTIVE SUMMARY.....	XI
LIST OF FIGURES	XIV
LIST OF TABLES	17
GLOSSARY	18
1. INTRODUCTION.....	19
1.1 CONTEXT	19
1.2 PROJECT SCOPE AND GOALS.....	19
1.3 STAKEHOLDERS	20
1.4 DELIVERABLES	20
1.5 REPORT OVERVIEW	20
2. DOMAIN ANALYSIS	22
2.1 REAL-TIME SYSTEMS	22
2.2 REAL-TIME OPERATING SYSTEM (RTOS)	23
2.3 REAL-TIME DISTRIBUTED SYSTEMS	27
2.4 REAL-TIME STREAMING PROTOCOLS.....	30
3. PROBLEM ANALYSIS.....	33
3.1 INTRODUCTION	33
3.2 1-CAMERA SYSTEM	33
3.3 2-CAMERA SYSTEM	41
3.4 SUMMARY POSSIBLE PROBLEMS.....	41
4. SYSTEM REQUIREMENTS AND WORK PLAN	43
4.1 FUNCTIONAL REQUIREMENTS	43
4.2 EXTRA-FUNCTIONAL REQUIREMENTS	45
4.3 WORK PLAN.....	46
5. FEASIBILITY ANALYSIS	47
5.1 INTRODUCTION	47
5.2 REAL-TIME LINUX INVESTIGATION	47
5.3 MULTI-MEDIA PLAYER SELECTION (VLC).....	49
5.4 SYSTEM HARDWARE AND SOFTWARE ORGANIZATION.....	50
5.5 INITIAL EXPERIMENT	50
5.6 PROBLEMS AND SOLUTIONS	62
5.7 SOLUTION SPACE	66
6. SYSTEM ARCHITECTURE AND DESIGN	67
6.1 INTRODUCTION	67
6.2 SYSTEM ARCHITECTURAL	67
6.3 SYSTEM DESCRIPTION	69
6.4 LOGICAL VIEW	71
6.5 PROCESS VIEW	83
6.6 DEPLOYMENT VIEW	86
7. IMPLEMENTATION AND PROBLEMS.....	90
7.1 TASK DIVISION AND SYNCHRONIZATION	90
7.2 SYSTEM INTEGRATION AND INTERFACES.....	93

7.3	IMPLEMENTED UNITS AND RESULTS.....	95
7.4	MAIN SOFTWARE PROBLEMS	99
8.	CONCLUSION.....	102
8.1	RESULTS AND CONCLUSIONS	102
8.2	RECOMMENDATION AND FUTURE WORK.....	103
9.	PROJECT MANAGEMENT	105
9.1	PROCESS	105
9.2	PLANNING AND TRACKING	105
9.3	RISK MANAGEMENT.....	106
9.4	RETROSPECTIVE	107
	BIBLIOGRAPHY.....	109
	APPENDIX-A.....	112
	DEBUGGING	115
	ABOUT THE AUTHOR	118

List of Figures

Figure 1-Simple View of Real-Time Systems	22
Figure 2-Throughput-Latency Relation	23
Figure 3-Model of Periodic Task	24
Figure 4-VDG Camera	25
Figure 5- μ C/OS-II Task States	25
Figure 6-Low Quality Image (a), High Quality Image (b)	26
Figure 7-Schematic View of Distributed System	27
Figure 8-Real-Time Distributed System End-To-End Timing	27
Figure 9-FPPS Illustration	29
Figure 10-FPNS Illustration, Legend is Same as Figure 9	29
Figure 11-Hierarchical Scheduling	30
Figure 12-Real-Time Protocols Network Layer Structure	30
Figure 13-RTSP Operations	31
Figure 14-Overview RTSP Request via Browser	31
Figure 15-Overview of Original System Setup with 2-Camera	33
Figure 16-Single Camera	34
Figure 17-Conceptual View of Distributed Video Processing	35
Figure 18-Application High-Level Overview of Camera	36
Figure 19-(A) Abstract Hardware View of Camera, (B) Data Flow on Camera ..	36
Figure 20-How Encoded Buffer is filled	37
Figure 21-Encoded JPEG Packet with Network Protocol Headers	38
Figure 22-Conceptual View of Video Processing without Streaming	38
Figure 23-Conceptual View of Video Processing with Streaming	38
Figure 24-Packetizing Illustration	39
Figure 25-Two Cameras are Connected to PC	41
Figure 26-2-Camera System Network Problem	41
Figure 27-Standard 2.6 Linux Kernel With Preemption	48
Figure 28-Command to check whether RTOS is correctly Patched or Not	49
Figure 29-General Format of Command that turns Non-Real-Time Task into Real-Time Task	49
Figure 30-Camera System Software Organization	50
Figure 31-Caching Experiment: Caching Size = 1000 Milliseconds	52
Figure 32-Caching Experiment: Caching Size = 0 Milliseconds	52
Figure 33-WireShark Output Example	53
Figure 34-Frame Rate and Number of Packet per Frame of Image at Different Quality Levels for Average Case Measurements	54
Figure 35-Total Number of Packets per Second in Different Qualities for Average Case Measurements	55
Figure 36-Task Execution Time Period in Worst-Case	55

Figure 37-Worst-Case Image consisting Lots of Strips, Resolution 400x400.....	56
Figure 38-Test Case Setup Ingredients.....	56
Figure 39-Frame Rate and Number of Packet per Frame of Image at Different Quality Levels for Worst-Case Complexity Measurements.....	58
Figure 40-Total Number of Packets per Second in Different Qualities for Worst-Case Complexity Case Measurements.....	58
Figure 41-Number of Frames for Each Quality Level in 1 Second.....	59
Figure 42-Time Measurements for Different Quality Level for Worst-Case Complexity Measurements (in 1 Second).....	60
Figure 43-Time Difference between Sequential Two Frames at Quality Level 50.....	60
Figure 44-Arrival Time of Received Packets at Quality Level 50.....	61
Figure 45-Processing and Transfer Time in between (Millisecond).....	62
Figure 46-1-Camera System, with 1 processor the state of the processor and the network.....	63
Figure 47-Problem-3: Inadequate Bandwidth.....	65
Figure 48-2-Camera System: 2 Cameras Stream Frame and Overlap Occurs, Same Legend in Figure 47.....	65
Figure 49-Overall System Architecture.....	67
Figure 50-System Layer Architecture Component View.....	69
Figure 51-Communication between the system nodes.....	70
Figure 52-Architectural Reasoning Diagram.....	70
Figure 53-Txt File contains Information gathered from Worst Case Measurements, Explained in Feasibility Analysis Chapter.....	72
Figure 54-Storage Unit contains Information gathered from Worst Case Scenarios in Advance, See Chapter 5.....	72
Figure 55-Message Sequence that Admission Control Unit accepts Connection from Camera Application.....	73
Figure 56-Admission Control Unit deny Connection.....	73
Figure 57-Connection Existence Check for Registered Cameras.....	75
Figure 58-Resource Enforcement Unit applied within Camera Application.....	76
Figure 59- Possible Scenario for Preventing Packet Overlap on Network.....	77
Figure 60-Giving Start Delay is processed within Camera Application.....	81
Figure 61-Time Unit Difference Representation on Cameras and on PC.....	83
Figure 62-Interaction Overview Among VLC, Camera Application and AC.....	85
Figure 63-Components Deployment and Architectural Decision Diagram.....	89
Figure 64-Video Task divided into Two Sub-Tasks.....	90
Figure 65-Video Task Functionality divided into Two Sub-Functions.....	91
Figure 66-Original Application Function Parameter Transfer.....	91
Figure 67-Message Synchronization between Tasks.....	91
Figure 68-Task Synchronization and Message Synchronization Conceptual View.....	92
Figure 69-Interface between System Nodes.....	93

Figure 70-Implementations within Camera.....	93
Figure 71-Admission Control and Bandwidth Availability in PC	94
Figure 72-Admission Control	94
Figure 73-Bandwidth Availability	95
Figure 74-AC-Management Protocol Communication with Camera Application.....	96
Figure 75-Camera Application Response to AC.....	96
Figure 76-Connection Availability Checking	97
Figure 77-Enforcement Units Integration to Camera [18]	97
Figure 78-Overlap Scenario	99
Figure 79-Incomplete Received Frame	100
Figure 80-Unhandled Exception Error	100
Figure 81-Weekly Meeting Presentations	105
Figure 82-Microsoft Office Project tool, iterative planning.....	106
Figure 83-Microsoft Office Excel Milestone Trend Analysis.....	106
Figure 84-CatapultEJ2-Ethernet –to-JTAG device: Yellow Wires are called Flying Leads, and Black-Green Head is called JTAG Print Head; JTAG Print Head is plugged to Camera.....	115
Figure 85-Connection between Nodes: Camera, PC and Catapult.....	116
Figure 86-Message Sequence among PC-Catapult-Camera: Camera Application Upload and Streaming.....	117

List of Tables

Table 1-Abbreviations and Descriptions	18
Table 2-List of Stakeholders and Expectations	20
Table 3-List of Deliverables	20
Table 4-Timing Attributes Description.....	24
Table 5-RTSP Streaming Commands.....	30
Table 6-Camera Platform Hardware Specifications	34
Table 7-Hardware Specifications	35
Table 8-Physical Restriction System Cause Problems	41
Table 9-RTOS Functional and Nonfunctional Requirements and Rationalities ...	43
Table 10-Multi-Media Player Functional Requirements	44
Table 11-List of Initial Experiments and reasoning.....	51
Table 12-Message Density on Network.....	57
Table 13-Problem-1 and Solutions	62
Table 14-Problem-2 and solutions	63
Table 15-Problem-3 and Solutions	64
Table 16-Additional Aspects for Problem_3	65
Table 17-Admission Control Scenarios and Actions	74
Table 18-Bandwidth Availability Unit Scenarios and Actions.....	74
Table 19-Resource Enforcement Unit Design Decision	75
Table 20-The Delay Unit Scenarios and Actions	77
Table 21-Prevent Overlap Design Decision	78
Table 22-Prevent Overlap Solution Approaches	78
Table 23-Delay At Once, Tasks State Design Decisions.....	78
Table 24-Video Task Division and Buffering	79
Table 25-Delay Unit Sequence Diagram Items	79
Table 26-Components Deployment Design Options	86
Table 27-Summary of Deployment Design Decision	87
Table 28-Functionalities and Interfaces within Camera Application	93
Table 29-Functionalities within Admission Control.....	94
Table 30-Delay At Once Abbreviations	99
Table 31-Most Important Identified Project Risks	107
Table 32-Real-time Operating System Criteria	112
Table 33-Selected RTOS	114

Glossary

Table 1 shows the abbreviations and descriptions; it is alphabetically ordered.

Table 1-Abbreviations and Descriptions

<i>Name</i>	<i>Description</i>
AC	Admission Control
BA	Bandwidth Availability
CPA	Camera Application
CS	Camera System
FPPS	Fixed Priority Preemptive Scheduling
HSF	Hierarchical Scheduling Framework
OMECA	Optimization of Modular Embedded Computer-vision Architectures
OOTI	Ontwerpers Opleiding Technische Informatica
RTNCS	Real-time Networked Camera System
RTOS	Real-time Operating System
SAN	Software Architecture and Networking Group

1. Introduction

This chapter presents the context and the goals of the current project. This is followed by a brief analysis of the stakeholders and deliverables. The chapter concludes with an overview of the structure of this report.

1.1 Context

Point-One is an open association of the high-tech industry and knowledge institutes in the Netherlands aims at the research & development of nano-electronics, embedded systems, and mechatronics.(1) The association funds projects such as the Optimization of Modular Embedded Computer-Vision Architectures (OMECA) project.

One of the technical objectives of the OMECA project is the invention of *new* technologies and tools for automated optimization of the design of adaptive networked real-time embedded systems with respect to multiple and contradictory extra-functional properties, e.g., performance, robustness, reliability and power consumption. (2)

The OMECA consortium consists of three enterprises, i.e., Prodrive, Gatsomer, and Cyclomedia and two universities, i.e., University of Leiden and Eindhoven University of Technology.

TU/e has many academic departments and groups, and the System Architecture and Networking (SAN) group is one of them. The SAN group studies parallel and distributed systems with an emphasis on resource constrained *networked embedded system* and focuses on distributed media systems, wireless sensor networks, automotive electronics, and more recently on lighting domains. (3)

The SAN group investigates real-time aspects of multi-media processing systems in the surveillance domain. It has developed and implemented a two-level Hierarchical Scheduling Framework (HSF) for a single processor using fixed-priority preemptive scheduling (FPPS), see section 2.3.1.2 for an explanation of HSF and section 2.3.1.2 for an explanation of FPPS. An HSF allows the developer of each real-time application to validate the schedulability of the application, independent of other applications. (4)

Moreover, the SAN group investigates application and system mode changes, in particular, related to changing memory requirements in streaming applications running on a single-processor platform. A system mode change is an overall change in the *allocation* of resources to applications (tasks). An application mode change is a change in the *requested* resources for that application.

The aim of the SAN group is to extend this work to networked devices (such as cameras) and to provide a predictable networked system for real-time environments as part of the OMECA project. Furthermore, the goal of this project is to implement a demonstrator of such a system for the OMECA project.

1.2 Project Scope and Goals

The project addresses a real-time networking setup with video play-out as the running example. The main focus of the report is on the results of the problem analysis and the design process, from both a technical as well as managerial point of view.

In the project proposal report, the goals are given as follows:

- Have a distributed platform, supported by a Real-Time Operating System (RTOS) on each node; the setup consist of two cameras and a PC. These cameras transmit streams to a PC.
- Have an example application that shows resource management in a distributed context, two cameras and a PC.

- Have a protocol for communication between the real-time Kernels which should enable the integration of real-time communication and distributed control in order to admit system-wide decisions.

1.3 Stakeholders

This is the author's final project for the Software Technology program of the Stan Ackerman Institute (SAI). The main stakeholder of this project is the SAN group, who is a partner in the OMECA project. Prodrive is one of the other partners in the OMECA project and also provides cameras for the current project.

Table 2 presents the roles of the most important stakeholders and expectations.

Name	Roles	Expectations
Ad Aerts	General director of OOTI SAI program.	To have a technical report which contains system architecture, system design and project management written in decent English.
Johan Lukkien	Scientific director of OOTI SAI program.	To have a technical report which contains the system architecture, system design explained in decent English.
	OMECA project group leader.	To have proper investigation of alternatives related to project goals within the project scope.
Antonio Liotta	OMECA project researcher.	To have a prototype that demonstrates the resource reservation and real-time scheduling in the distributed network system.

Other relevant stakeholders include software engineers and software designers who need an extensible and understandable design in order to implement new features. Deployment managers are also a relevant stakeholder due to the fact that they need to deploy the current project and integrate it with the distributed networked systems' components. Therefore, the project should be well documented.

1.4 Deliverables

In this section, a set of deliverables of the current project are defined according to the SAN group and TU/e needs. This set is presented in Table 3.

Deliverables	Description
Prototype	This includes software, which demonstrates the resource management, and distributed scheduling in the real-time, distributed video processing application.
Technical Report (this document)	This document describes the design and implementation of the software, the feasibility study that includes all the problems encountered while combining standard and nonstandard technologies, as well as the results and conclusions of the current project.
Supporting Documents	These include project management, analysis and architecture documents, and a user guide.

1.5 Report Overview

- **Chapter 2 Domain Analysis:**
This chapter provides information on the terminologies that will be used in the following chapters. In the first part, real-time systems are explained

briefly. The second part explains the related software components: real-time operating systems and the video processing application. Then real-time co-ordination in distributed systems is explained and finally appropriate protocols for the real-time streaming are given.

- **Chapter 3 Problem Analysis:**

This chapter presents an in-depth analysis of the camera system, its setup and problems. The camera system setup originally has two cameras and one PC. This chapter is divided into two main sections: camera system setup with one camera (1-camera system) and with two cameras (2-camera system). In the 1-camera system, hardware and software components are examined and their associated problems are given. Then, additional problems in the 2-camera system are pointed out.

- **Chapter 4 System Requirements and Work Plan:**

This chapter presents all the functional and extra-functional requirements along with the rationale for each of them. Some of the requirements are given in the project description report. Nevertheless, new requirements are derived from the given requirements such as the choice of RTOS and multi-media player.

- **Chapter 5 Feasibility Analysis:**

This chapter starts with a comprehensive investigation of the real-time operating system and the multi-media player. Then the initial measurements of the camera setup are given. Finally, the solution space is described.

- **Chapter 6 System Architecture and Design:**

This chapter provides a comprehensive architectural overview of the current project. It presents the architectural of the system; it defines the main components and the interaction between them by making use of several scenarios. Two diagram notations are used: a new proposal for architectural knowledge management (5) and UML.

- **Chapter 7 Implementation and Problems:**

This chapter gives a detailed explanation of the component implementation. First of all, the video task division and synchronization are explained. Then a complete system description is given and the interfaces and components are explained. This is followed by a description of the Management Protocol and explanation of the delay component. The chapter is concluded by stating the software problems on the camera application and the possible resolution of these problems.

- **Chapter 8 Conclusion:**

- This chapter presents the overall conclusions of the “A Real-Time Networked Camera System” project. Section 8.1 states all the conclusions, section 8.2 provides some recommendations and suggests a possible future extension of the current project.

- **Chapter 9 Project Management:**

This chapter introduces the various issues that are relevant to project management. The process used to manage the project is described in the first part. Other related subjects such as Breakdown structure, Milestone Trend Analysis, and Risk management are also presented in this section. A short retrospective of the project encloses the chapter.

2. Domain Analysis

This chapter provides information on the terminologies that will be used in the following chapters. In the first part, real-time systems are explained briefly. The second part explains the related software components: real-time operating systems and the video processing application. Then real-time coordination in distributed systems is explained and finally appropriate protocols for the real-time streaming are given.

2.1 Real-time Systems

A Real-time System (RTS) is a system in which the correct operation depends not only on the functional correctness of computed values but also on the time during which these results are produced. RTSs cover a broad spectrum from very simple devices to very complex machines which are involved in the gathering and processing of data, and providing timely responses. Response time is the distinguishing factor between real-time and non-real-time systems. The design of non-real-time systems aims to have maximum throughput whereas the aim of real-time systems is to *guarantee* that all the tasks are processed in a given time. Figure 1 shows a simple view of the real-time systems domain. (6)

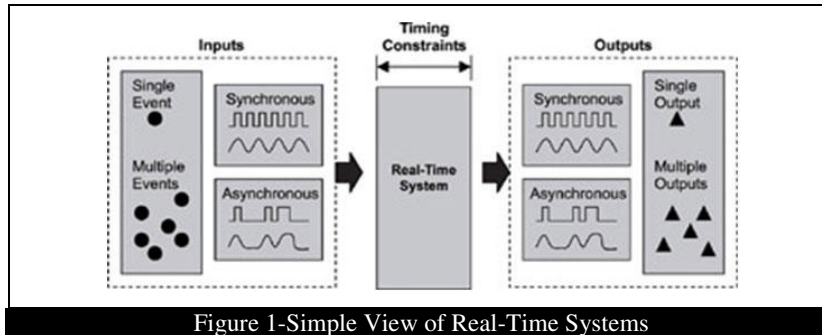


Figure 1-Simple View of Real-Time Systems

There are two underlying objectives in RTS design: predictability and low latency.

- *Predictable* means that it should be possible to show, demonstrate, or prove that requirements are met subject to assumptions, such as concerning failures and workloads. In case of static environments, the overall system behavior can be predicted. For dynamic environments, however, it is hard to predict.
- *Latency* means the sojourn time of the packets in the buffer (time spent in the buffer) that causes a delay between the sending and receiving. Figure shows the latency.

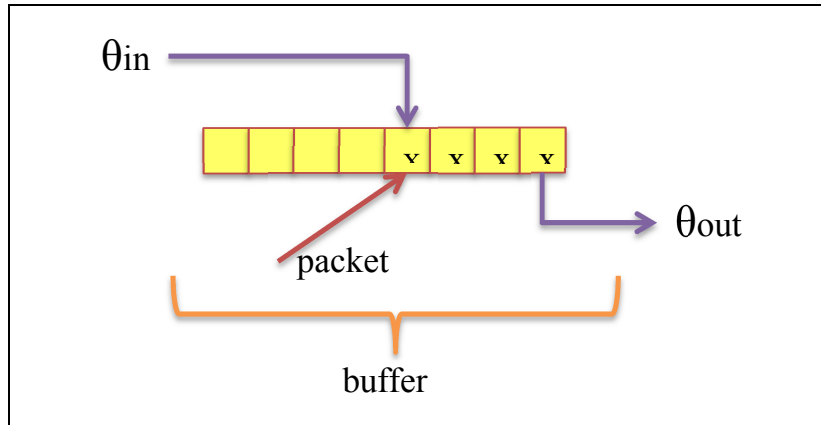


Figure 2-Throughput-Latency Relation

If in the long run the arrival rate $\theta_{in} > \theta_{out}$, the departure rate, the buffer eventually fills up completely and packets will be overwritten causing the application to suffer packet loss. If in the long run $\theta_{in} < \theta_{out}$, the θ_{out} process will eventually run out of work and suffer due to starvation. So on average $\theta_{in} = \theta_{out}$ is the desired situation: there is balance between the input rate θ_{in} and the output rate θ_{out} . For example: When assuming that the average occupancy of the buffer is 4 and $\theta_{in} = \theta_{out} = \theta$, the latency is “ $4/\theta$ ”.

The tolerance for the latency depends on the system. Real-time systems are classified in three types: hard, firm, and soft RTSs. The distinction between them is based on the flexibility with which they handle time constraints.

- A *hard real-time system* is required to produce its results within certain predefined time bounds. An example of a hard real-time system is a flight control system.
- A *firm real-time system* is associated with some predefined deadlines before which it is required to produce results. However, unlike a hard real-time task, even when a firm real-time task does not complete within its deadline, the system does not fail. An example of a firm real-time system is a video conferencing system.
- A *soft real-time system* is a system in which deadlines are important but which will still function correctly when deadlines are occasionally missed. An example of a firm real-time system is an online database.(7)

2.2 Real-Time Operating System (RTOS)

One major component in the design of real-time systems is the Operating System (OS). The OS must provide basic support for guaranteeing real-time constraints, supporting fault-tolerance and distribution, and for integrating time-constrained resource allocations and scheduling different resource types: sensor processing, communications, CPU, memory, and other forms of I/O. (8)

A real-time OS (RTOS) supports real-time applications (RTAs). RTAs have the requirement to meet task deadlines in addition to the logical correctness of the results. RTAs can be both embedded applications and desktop applications. In most cases, RTAs are embedded on customized devices which can be used for special purposes; in this project Prodrive (OMECA partner) provides two cameras and these cameras are customized specifically for video capturing and processing with the installation of the μ C/OS-II real-time kernel (Version 2.88).

An RTOS allows RTAs to be designed and expanded easily. The use of an RTOS simplifies the design process by splitting the application code into separate tasks.

It allows RTA designers to make better use of system resources by providing valuable services such as semaphores, mailboxes, queues, time delays and time outs.(9) For better understanding of the report, some RTOS terminology is briefly explained in Figure 3 and Table 4.

Generally, real-time tasks are categorized as periodic or aperiodic. *Periodic tasks* are initiated on regular time intervals and have to be executed within that time interval. *Aperiodic tasks* occur randomly, i.e., these tasks have irregular arrival times. (10) Timing attributes and necessary terms (task, job) are described in Table 4 and they are visualized in Figure 3. (11)

Table 4-Timing Attributes Description	
Attributes	Definition
Task	Consists of a series of instructions in response to some event(s).
Job	Instance of a task.
Φ_i	Phase (offset) of task i , release time of its first instance.
Relative Deadline	Length of time between release time and absolute deadline.
Absolute Deadline	Time by which the execution of a job is required to complete.
Preemption	Task can be interrupted and processor can be assigned to another job at any time, for more detailed information please refer to section 2.3.1.1
Response Time	Length of time between the release and completion.
Completion Time	Time in which the execution of a job is completed.
Execution Time	Maximum length of time a task needs to execute.
Period	Minimum time between the releases of a job.
Release Time	Time at which a job becomes available for execution.
WCET	Worst-Case Execution Time of a job.

The visualization of the terms is given in Figure 3.

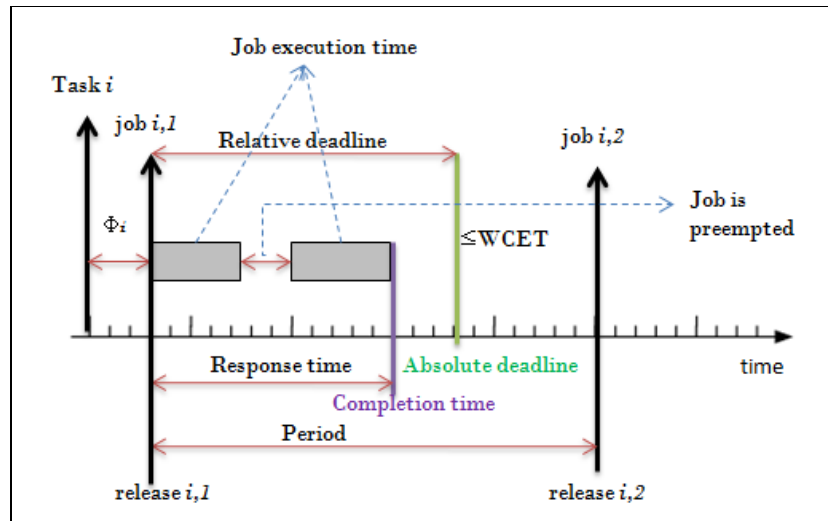


Figure 3-Model of Periodic Task

The following two sections contain information on the real-time kernel and the camera application.

2.2.1 Real-Time Kernel μ C/OS-II

μ C/OS-II is a real-time preemptive OS designed for embedded systems. It is delivered with a complete ANSI C source code and documentation. It is developed with portability in mind; so various ports to different CPU

architectures are available. For example, the *Stretch* processor (Figure 4) is the processor that is used in the camera as used in current project.



Figure 4-VDG Camera

μ C/OS-II is composed of several components and each component consists of at least one module. The most important component is the *kernel component*. It provides basic OS functionalities: semaphores, event flags, mutual exclusion, message boxes and queues for synchronization. Task, time & timer management, and fixed size memory block management are also provided by the μ C/OS-II real-time kernel. In addition, it provides network modules in order to interact with networks, e.g. HTTP, TCP/IP and UDP protocols.(12)

μ C/OS-II is capable of managing up to 250 application tasks but in this specific project μ C/OS-II only needs to provide 64 tasks. The most important tasks are: *Core Task*, *Idle Task*, and *Timer Management Task*. The *Core Task* starts the other tasks, the *Idle Task* receives the processor time when there is no task running and when it is doing nothing, and the *Timer Management Task* manages the timer inside the μ C/OS-II. It counts down the specified time and when the time is up, it executes its assigned functions.

Each task has a unique priority. Task priority is inversely correlated with the scheduling order of the task, i.e., the highest prioritized task has the smallest priority number and is scheduled first.

Figure 5 shows the life cycle of tasks in μ C/OS-II. When a task is in the *Ready state*, the scheduler (dispatcher) dispatches the highest prioritized task which is in the *Ready* queue. The scheduler in μ C/OS-II processes the tasks using the fixed priority preemptive scheduling (FPPS) algorithm. So, when during execution a higher priority task becomes ready, the higher priority task will preempt the running task. If during execution a task is unable to continue, perhaps due to a semaphore, it will be placed in the *waiting group* and the next highest priority task will be selected by the scheduler to run.

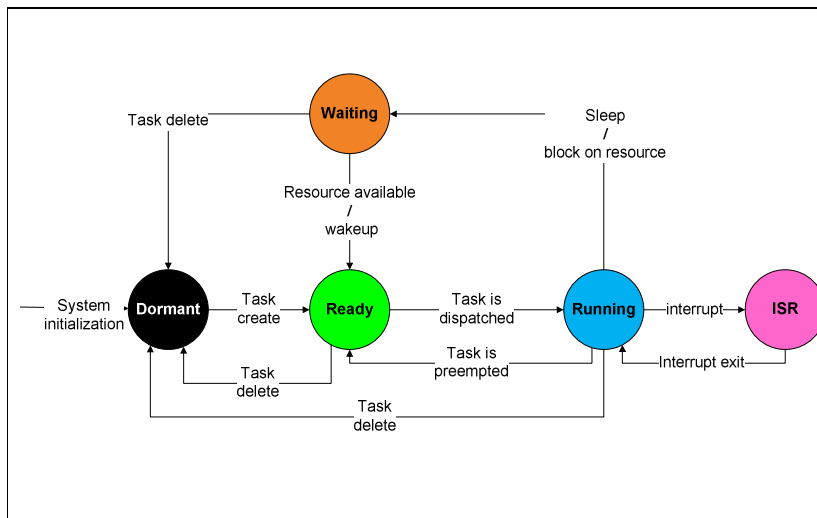


Figure 5- μ C/OS-II Task States

Besides tasks, μ C/OS-II also manages Interrupt Service Routines (ISRs). These are routines that can interrupt any task, at any time, to perform a specific set of actions. Usually, these are important actions that need to be handled quickly.

2.2.2 Video Processing

Video processing is used as an example RTA in the current project. A video is a series of images and a substantial part of the video processing is done on an image by image basis. There are two major motives for the image processing. The first motive is the improvement of pictorial information for human perception which leads to enhancing the image quality so that it will have a better look. The second motive is efficient storage and transmission; for this purpose the individual images are compressed.

Video communication usually relies on compressed video streams. Transmission of uncompressed (raw) video streams is impractical when the transmission capacity is limited. Storage of raw video streams is impractical as well, when the memory medium capacity is limited. Excessive bandwidth is needed for both the communication channel and storage devices. Processing speed of the tasks and memory limitations often impose serious constraints on transmission rates. (13)

Over the last decade, a number of compression methods and video formats have been released by international organizations. Two major compressions standards are H.26X and MJPEG which are both used for high quality video applications. MJPEG is the video format used in this project. MJPEG is simply defined as encoding each individual image separately in the video sequence using the JPEG compression. The JPEG standard is given and a good introduction can be found on. (14)

For image specifications, there are two important metrics; resolution and quality:

- *Resolution* refers to a number of pixels in an image. It is given generally in height and width, such as 1920x1080 which means 1920 pixels for width and 1080 pixels for height.
- *Quality* refers to the perceptible visual quality of an image. By using lossy coding methods, fewer bits are needed for representing the image at the expense of a loss in quality.

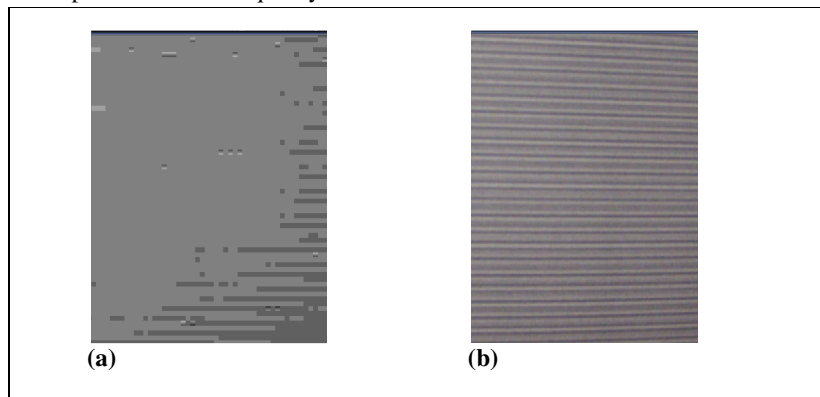


Figure 6-Low Quality Image (a), High Quality Image (b)

Figure 6 displays the result of processing a single captured image with two different quality levels: Figure 6-(a) presents low quality; the image is almost unrecognizable, Figure 6-(b) presents high quality; the lines are discernable. The images are captured from the image given in Figure 37.

2.3 Real-Time Distributed Systems

The concept of a distributed system is based on (15): “A Distributed system is the hard- and software of a collection of independent computers that cooperate to realize some functionality.” Therefore, the camera system used in this project is a distributed system because it consists of two or more distinct cooperating machines to achieve video capturing and playback functions.

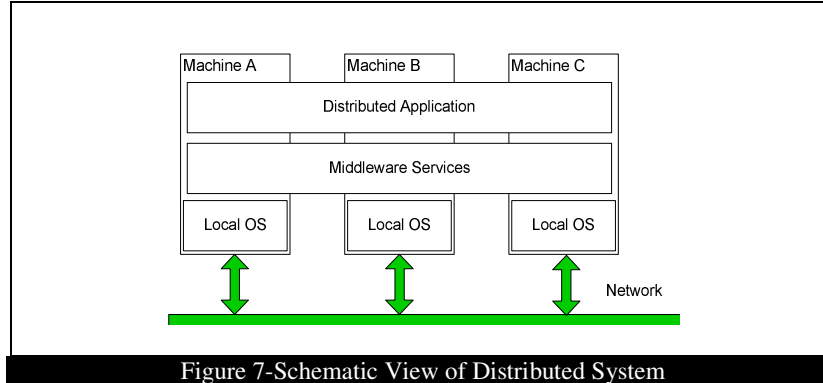


Figure 7-Schematic View of Distributed System

Figure 7 shows a schematic view of a possible organization of a distributed system. There are multiple machines which are linked to each other via a network. There is a single application which has multiple tasks that are processed on different machines. Furthermore, there are middleware services on each machine that provide the coordination for task processing within the distributed system.

There are two main motivations for designing a distributed system:

- The problem statement is inherently distributed.
- It is chosen as part of the solution, in order to achieve certain extra-functional properties.

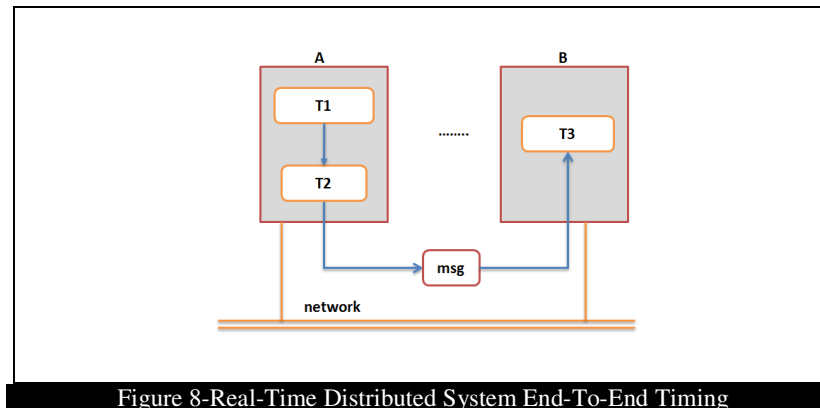


Figure 8-Real-Time Distributed System End-To-End Timing

Figure 8 shows an application consisting of cooperating tasks, labeled T (1, 2, and 3), in a distributed environment. Communication is both internal, between tasks on the same machine, and external, between tasks on different machines. Communication between the tasks on different machines is provided via messaging, it also shows the dependency between the tasks, e.g., T3 depends on T2 and T2 depends on T1.

In distributed RTSs, time constraints are applied to collections of cooperating tasks, and not only to individual tasks. Current timing must be accomplished under a single end-to-end timing constraint. The timing on the network should be considered as well.

In RTSs, resource demands often change dynamically over time and are not known a priori. Also resource availability may change over time. For these reasons RTAs need a capability to adapt to changing conditions in a way that does not violate their temporal requirements in an uncontrollable manner.(16)

2.3.1 Resource Reservation

Resource reservation implements the *temporal isolation of the resource*, which is the capability of a set of processes running on the same node without interferences concerning their temporal constraints.(17)

Resources are virtualized by resource reservation so that tasks cannot access a resource directly. Real-time tasks are guaranteed a requested share of such a virtualized resource. Various kinds of resources can be virtualized in such a fashion, such as disks, network (bandwidth), and processor (CPU) time. The following mechanisms are required in order to guarantee resource reservation.(18)

- **Admission:** establishing whether to accept a new request or not.
- **Scheduling:** scheduling the tasks according to their reservations and the admission policy.
- **Monitoring:** keeping track of the execution time used by applications.
- **Enforcement:** ensuring that tasks do not utilize more resources than they should.

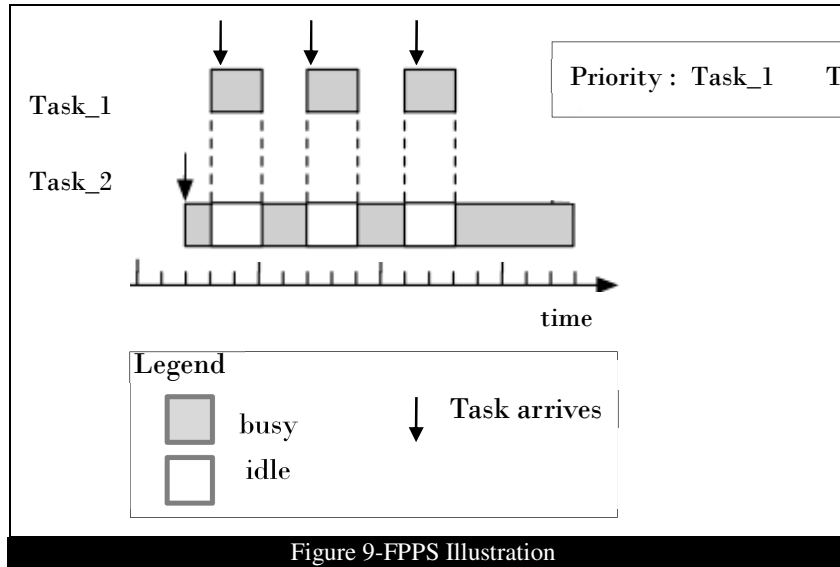
2.3.1.1 Scheduling

Scheduling is the process of deciding which task is granted access to a shared resource at a given time. Scheduling of real-time tasks is very different from general scheduling. Ordinary scheduling algorithms attempt to ensure fairness among tasks, progress for any individual task, and absence of starvation and deadlock (19). Two types of scheduling are discerned in this report. The first type is fixed-priority-scheduling (FPS) and the second type is dynamic-priority-scheduling (DPS).

There are three types of FPS: fixed priority pre-emptive scheduling (FPPS), fixed priority non-pre-emptive scheduling (FPNS) and fixed priority scheduling with deferred preemption (FPDS). In FPS, the priority of a task remains constant during the execution, while in DPS, as the name suggests, it may change dynamically during the execution of the task, according to the relative deadlines of other tasks.

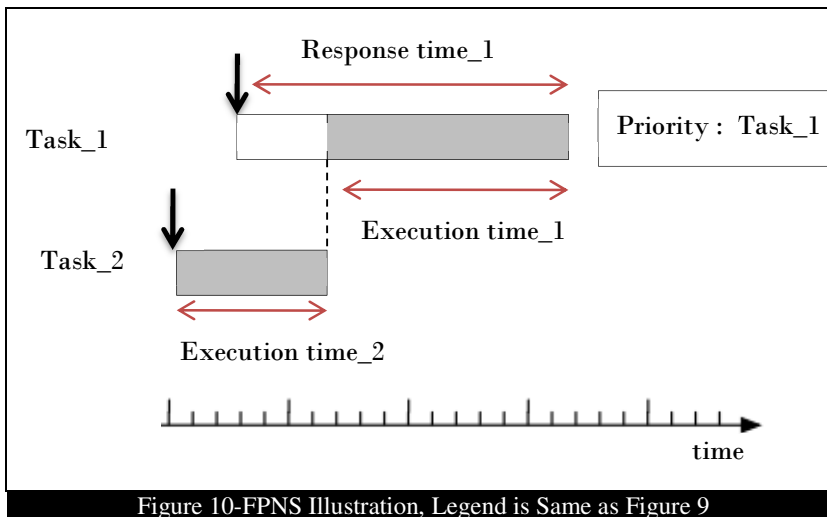
There are two types of DPS: earliest-deadline-first-scheduling (EDF) and least-slack-time-scheduling (LSTS). There are also two more scheduling methods for tasks that have the same priority; Round-Robin (RR) and First-In-First-Out (FIFO).

In preemptive scheduling, the currently executing task may be preempted, i.e., interrupted, if a more urgent task requests service.



In Figure 9, Task_1 has a higher priority than Task_2. When Task_1 requires using the resource even though it is used by Task_2 at that moment, Task_1 preempts Task_2 and starts to use the resource till it complete its job or is preempted by another high priority task.(20)

In non-preemptive scheduling, the currently executing task will not be interrupted until it decides on its own to release the allocated resources. Non-preemptive scheduling is reasonable in a task scenario where many short (compared to the time it takes for a context switch) tasks must be executed. (21)

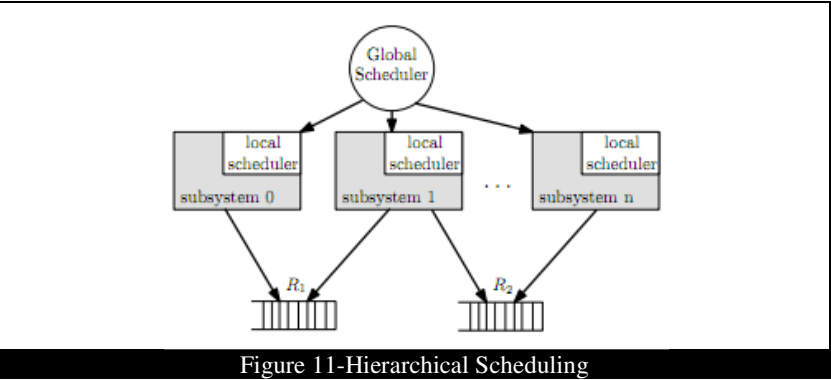


In Figure 10, Task_1 has a higher priority than Task_2. While Task_2 is using the resource, Task_1 requests the same resource. However, Task_1 is not allowed to use the resource till Task_2 completes its job.

2.3.1.2 Hierarchical Scheduling Framework

In Hierarchical Scheduling Framework (HSF), a system can be recursively divided into a number of subsystems that are scheduled by a global (system-level) scheduler. Each subsystem contains a set of tasks that are scheduled by a local (subsystem-level) scheduler. HSFs are inherently based on *virtualization* techniques (i.e. reservations), providing (temporal and spatial) *isolation* between applications, and are therefore an essential ingredient for robustness. (22)

Virtualization is the methodology of dividing the resources of a computer, e.g. processor, network, into multiple execution environments. Thus, it isolates the resources needed by an application from competing applications. Figure 11 depicts the *Hierarchical Scheduling* with an illustration of global-local schedulers, subsystems, and shared resources.(23)

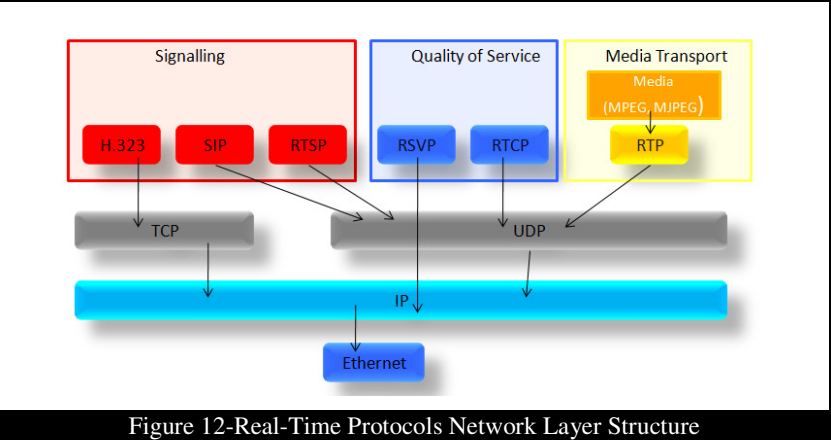


Tasks, located in arbitrary subsystems, may share logical resources.

2.4 Real-Time Streaming Protocols

In a network, connected machines communicate with each other using a variety of protocols. Since the project discussed in this report, is concerned with real-time video applications, we briefly explain the real-time streaming protocols.

A number of real-time protocols exist for the different needs of multimedia streaming over the network as shown in Figure 12. These protocols are differentiated, based on their application fields. In Figure 12, real-time protocols are illustrated in the structure of network layers.(24)



- *The Real-time Streaming Protocol (RTSP):*
 - Allow a media player to control the transmission of a media stream, i.e. pause/resume, repositioning of playback, fast forward and rewind.
 - Retrieve a media object from a server.
 - Invite a server to add a media object in an existing session.

Table 5 describes the RTSP streaming commands.

Table 5-RTSP Streaming Commands	
Message	Description
Options	Get available methods, e.g. DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE.

Describe	Get a (low level) description of the media object.
Setup	Establish the transport.
Play	Start playback, reposition.
Pause	Halt delivery, but keep state.
Teardown	Remove state, session.

Figure 13 shows the possible RTSP messaging sequence between the media server and client.

One of the requests from the PC is the RTSP SETUP message. It contains a local port for receiving RTP data. Hence, the server knows which port it will use to send RTP packets to the client side.

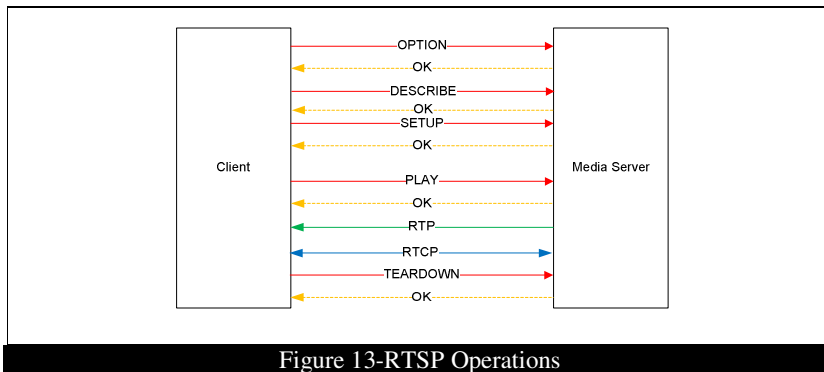


Figure 13-RTSP Operations

- *The Real-time Transport Protocol (RTP)* is an internet protocol which defines a standardized packet format for delivering audio and video over IP networks. RTP runs on top of the User Datagram Protocol (UDP). If streamed packets get lost, they are not be retransmitted. It is more important to transmit the stream in an RT fashion. For this reason, RTP runs on top of UDP, a connectionless protocol. TCP is less suitable for real-time protocols because of its retransmission scheme.(25)
- *The Real-time Control Protocol (RTCP)* provides the periodic transmission of control packets to all participants in the session using the same distribution mechanism as the data packets, in this case RTP. It uses a port number which is related to the RTP port number, e.g. if the RTP port number is n , the RTCP port number is $(n+1)$. (26)

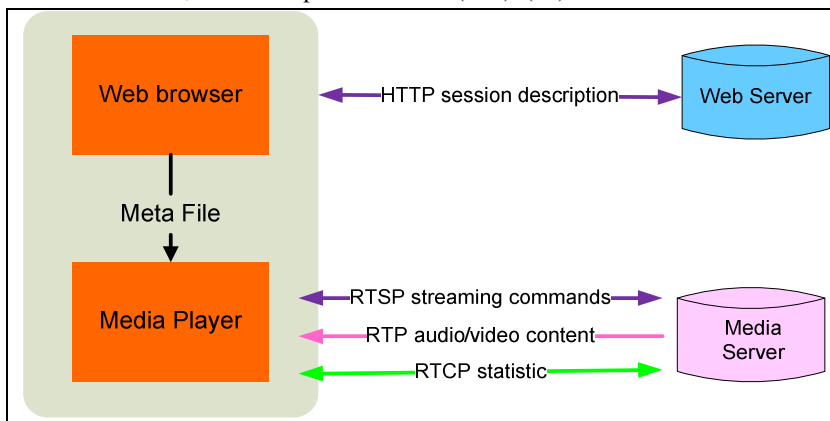


Figure 14-Overview RTSP Request via Browser

Figure 14 illustrates the real-time streaming protocol usage by a web browser equipped with a media player plug-in. When a user requests RTSP streaming via a web browser, the plug-in requests the RTSP streaming from the streaming server.

In summary, this project focuses on the real-time distributed video processing. In this chapter the domain and the related terms were given. First of all, real-time systems and their terminology were defined. Then the ingredients for the RTS were given: RTOS and RTA. Subsequently, the distributed systems and the additional components were given. Also, the coordination of the distributed systems and appropriate protocols were explained.

3. Problem Analysis

This chapter presents an in-depth analysis of the camera system, its setup and problems. The camera system setup originally has two cameras and one PC. This chapter is divided into two main sections: camera system setup with one camera (1-camera system) and with two cameras (2-camera system). In the 1-camera system, hardware and software components are examined and their associated problems are given. Then, additional problems in the 2-camera system are pointed out.

3.1 Introduction

Recall that the goals of the project are:

- Have a distributed platform, equipped with a Real-Time Operating System (RTOS) on each node; the setup consists of two cameras and a PC. These cameras transmit streams to a PC.
- Have an example application that shows resource management in a distributed context, two cameras and a PC.
- Have a protocol for communication between the real-time Kernels which should enable integration of real-time communication and distributed control in order to admit system-wide decisions.

In order to realize the goals, the system shown in Figure 15 is used. The system setup is limited to two cameras and one PC. Two cameras are connected to the PC via a network switch and each component is connected to the network switch with an Ethernet cable as a Local Area Network (LAN). The system provides end to end communication between the nodes.

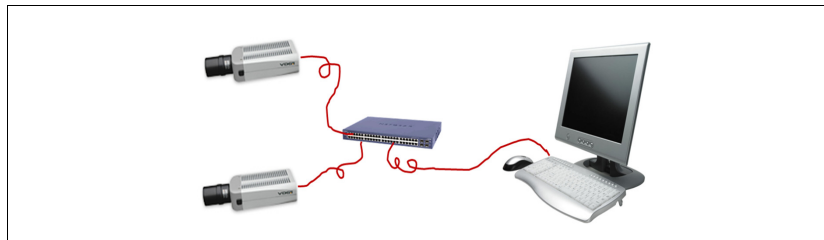


Figure 15-Overview of Original System Setup with 2-Camera

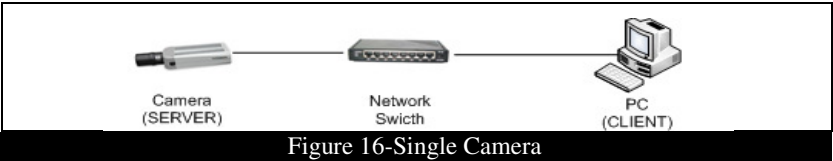
The description of the distributed system is divided into two main sections: 1-camera system and 2-camera system. The 1-camera system is the baseline for the 2-camera system. In the first section, the 1-camera system, the hardware, and software details are described. The physical constraints and the systems are examined in the second section; the 2-camera system is described with respect to the information which is given for the 1-camera system, in order to define the possible problems on the network.

3.2 1-Camera System

The components used in the 1-camera system are divided into two categories: hardware and software. First, each hardware component and the software running on the hardware components are explained. Thereafter, physical restrictions and possible problems are given.

3.2.1 Hardware

The hardware of the 1-camera system consists of one camera and one PC. They are connected to each other via a network switch. As shown in Figure 16, the setup consists of one camera transmitting a stream to a PC over the network.



3.2.1.1 Camera Hardware

The camera is a security camera designed by VDG security which is a commercial vendor from the surveillance domain. There are two dependent physical components in the camera: the *Sensor* board and the *Video Processor* board which are connected to each other via the *data port*. The Sensor board is used to *capture* raw video images and the Video Processor board is used to *encode* the video images and *transmits* them over the network. Table 6 gives the main specifications of two main parts of the camera.

Table 6-Camera Platform Hardware Specifications

Camera Platform		Hardware Component	Sp
	Sensor Board (P5MSB-A)	Sensor Board (P5MSB-A)	51
	Video Processor Board (VPA-PM)	Video Processor Board (VPA-PM)	2

3.2.1.2 PC and Network Switch Hardware

The PC is a common desktop PC; the network switch is used to join multiple machines together within one Local Area Network (LAN). Table 7 gives the main

specifications of the PC, the network switch and the Ethernet Cables that are used in the current project.

Table 7-Hardware Specifications	
Hardware Component	Specifications
Desktop PC	Core i-5 Intel processor 3.2GHz
	X86-64 architecture
	RAM 3,87 GB
Network Switch	100 Mbps
Ethernet Cable	100 Mbps full-duplex

The key component for the experimental setup is the network switch. It provides the communication between the camera and the PC by using an Ethernet cable. The network bandwidth available is 100Mbps.

The hardware of the 1-camera system is the physical constraint of this project. The speed of the processors, capacity of the bandwidth and size of the memory are as given in Table 6 and Table 7.

3.2.2 Software

Figure 17 shows the distributed video processing on the 1-camera system. The video application runs on the camera, it streams the compress video over the network to the PC. The processing of the compressed frame is finalized on the PC and the application on the PC decompresses and displays the frame.

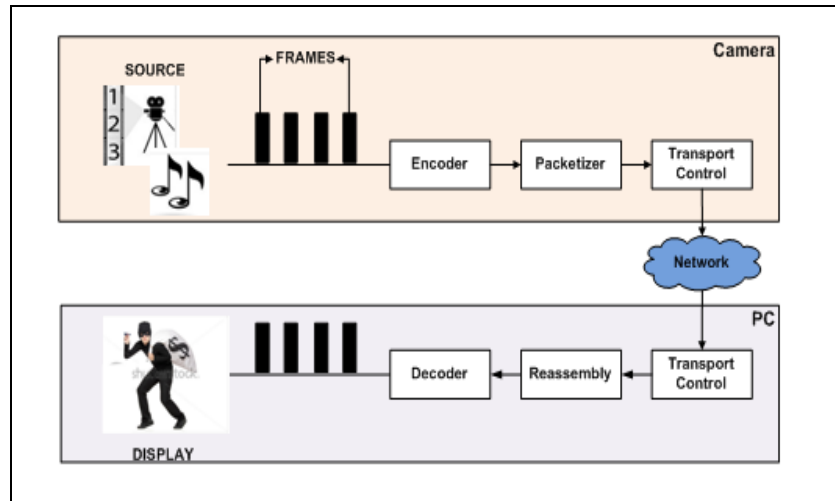


Figure 17-Conceptual View of Distributed Video Processing

3.2.2.1 Camera Software

On the camera, the μ C/OS-II real-time kernel is installed, see section 2.2.1. The video application on the camera is composed of two core applications: the S6SCP application and the S6AUX application. The S6SCP application is the main application and uses the S6AUX application to speed up the *encoding* of video. Figure 17 shows the application high-level overview: it is divided into three main layers: Application Layer, Kernel Layer and Hardware Support Layer. In Figure 17, arrows illustrate the communication between the components and modules. (27) The VPA-PM (Video Processor) board is supported by the VPA-PM library containing the functionality needed to initialize the components on the board. The VPA-PM library has a dependency on the Stretch (processor) BIOS (SBIOS) library.

The P5MSB-A (Sensor) board is accompanied by a library. The P5MSB-A library contains all the necessary functions and data structures for the operation of the sensor board. It is dependent on the library of the video processor board. (27)

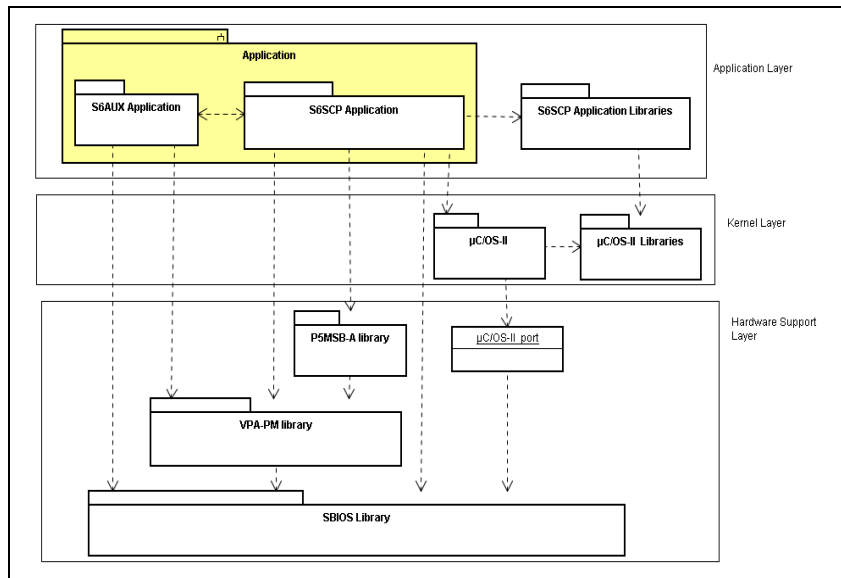


Figure 18-Application High-Level Overview of Camera

The application on the camera provides the following two services to the user:

- Video streams together with the audio and signaling streams, but these latter are ignored in this project.
- Method for changing the configuration of the camera via a website, such as quality, resolution, and brightness of the video image.

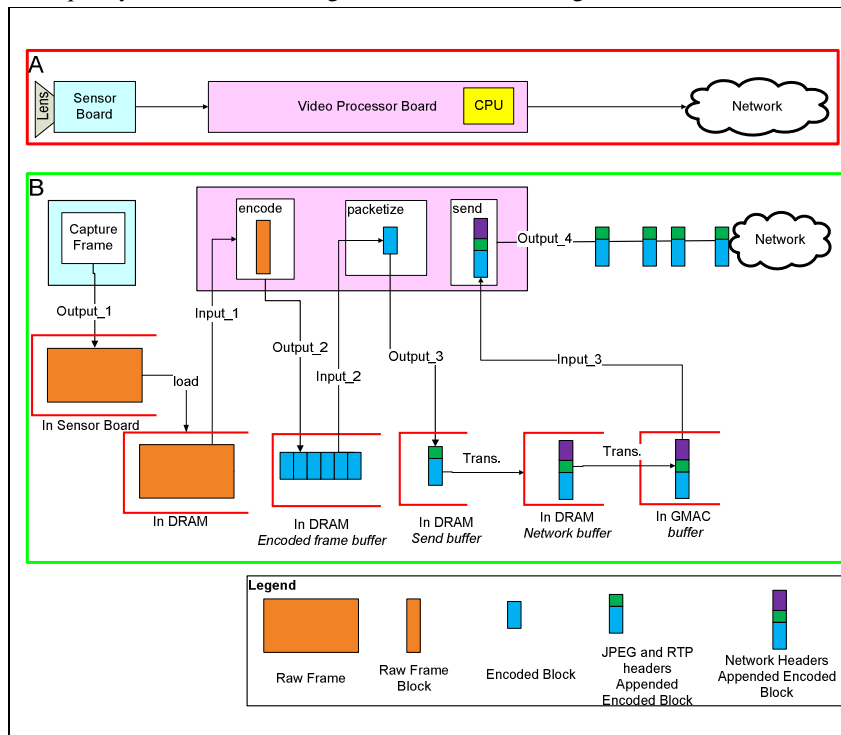


Figure 19-(A) Abstract Hardware View of Camera, (B) Data Flow on Camera

Figure 19-(A) shows the specific hardware components of the camera and Figure 19-(B) shows the process and data flow on the hardware components of the camera. It shows the data buffering on the hardware as well.

As long as the camera is turned on the Sensor board creates a 5 Mega pixel frame every 40 millisecond. Each frame is stored in the memory which is located on the

Sensor board; see Figure 19-B, Output_1. When a frame is stored in the memory, an interrupt is raised and the video task, which is part of the application on the video processor, is triggered to load the frame to the memory which is located on the Video Processor board.

The following two sections contain detailed information about the video processing. The video task is basically divided into three subtasks as shown in Figure 19-B. The purpose of the division is to explicitly explain their processes. The parts are shown in Figure 19-B: (I) encode, (II) packetize, and (III) send.

I. Frame Encode.

II - III. Frame Packetize and Send, they are explained together.

I- Frame Encode

A Raw Frame (RF) is captured by the Sensor board. In order to encode an RF, the encoding function needs some input parameters, such as RF (see Figure 19-B Input_1) quality, brightness, and sharpness. In the original setup these parameters have default values. However, the user can change these parameters and the resolution of the frame via the website. Whenever the parameters are changed, the changes are applied to the next RF.

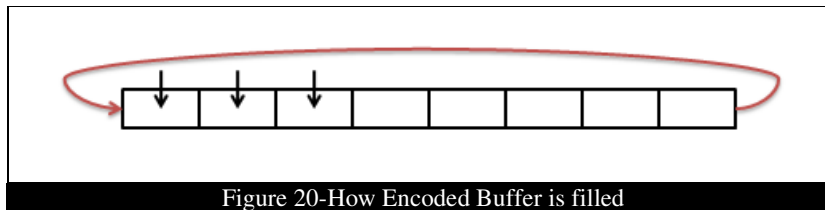
An RF is not encoded all at once; it is chopped into small pieces. Furthermore, these are encoded one after another, because the processor cache does not have enough space to process them all at once. The output of the encoding function is an Encoded frame (EF); see Figure 19-B Output_2.

The encoding process is applied till a grabbed RF is completely encoded. The size of the EF depends on two main parameters: the resolution of the RF and the quality level of the encoding function. Process time is proportional to both the frame and the quality level of the encoding function. For example; the processing time for the frame displayed in Figure 6-(b) is higher than that of the one displayed in Figure 6-(a), because it has higher quality.

If there is no connection request after encoding the frame, the new RF is grabbed and starts to be encoded.

II- Frame Packetize and Send

As long as there is a client connected to the camera, frames are packetized and sent. The EF is stored in the *encoded frame buffer* which is located in main memory. The encoded frame buffer is a single dimensional array and the size of the buffer is by default 1024 KB (this can be increased). Figure 20 represents the encoded frame buffer and is filled circularly. If the encoded frame buffer is larger than the remaining (free) buffer size, then the algorithm starts replacing the frames at the beginning of the buffer.



In order to transfer the encoded frame over the network it is chopped into small pieces, i.e. packets (see Figure 19-B Input_2). For each packet a 1400 bytes data block is grabbed from the encoded frame buffer, and JPEG and RTP headers are appended to the data block and stored in the *send buffer* which is located in main memory as well. Note that the packet, which is not complete yet because the required network headers still need to be added, is transferred to *network buffer*, located in the main memory. The required network headers are prepared and

added to this packet (Figure 19-B Input_3) and transferred to the *buffer* (GMAC, gigabit media access controller). Then the complete packet (Figure 19-B Output_4) is transferred over the network.

The network headers are UDP, IP and Ethernet protocol headers. This process is repeated until the end of the frame is reached. The size of the payload (the last packet is exceptional) and the headers are always the same.

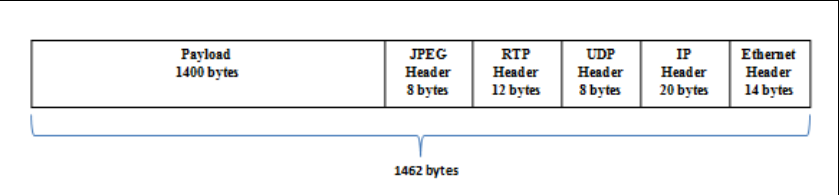


Figure 21-Encoded JPEG Packet with Network Protocol Headers

The number of packets transferred to the network depends on the size of the encoded frame. If the RF is encoded with high quality, the size of the encoded frame will be large. If the size of the EF is large, the number of packets transferred to the network will be high as well.

1. Physical Restriction on Camera Application

The sensor board produces frames at a rate of 25 FPS (frame per second), because it produces a frame every 40 millisecond, but the video task, part of the video application, is not as fast as the RF creation process. For this reason the frame rate of the EF-stream is less than the frame rate of the RF-stream. Moreover, the ratio FPS_{EF} / FPS_{RF} drops as the quality of encoding improves.

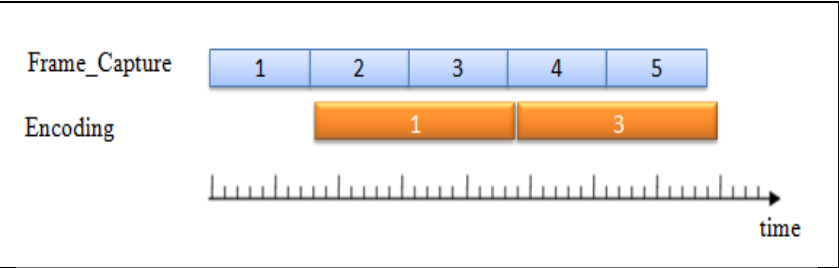


Figure 22-Conceptual View of Video Processing without Streaming

Figure 22 illustrates the video processing: video image capturing and encoding. Frame 2 is not encoded by the video task, because when the encoding of Frame 1 is completed, Frame 3 is already captured and Frame 2 is overwritten.

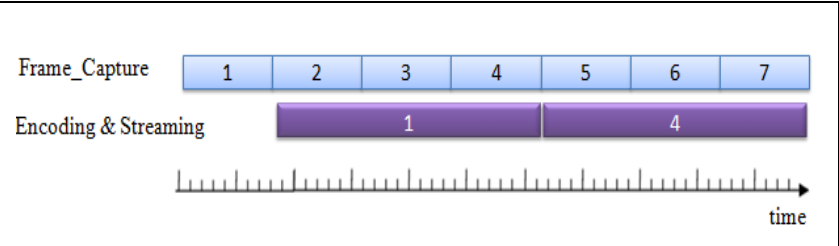


Figure 23-Conceptual View of Video Processing with Streaming

Figure 23 illustrates the video processing: video image capturing, encoding and streaming. Frames 2 and 3 are not encoded by the video task, because encoding the 1st frame and streaming over the network takes longer than capturing a frame.

In the current project, the video task is one complete task which contains encoding and streaming.

2. Physical Restriction imposed by Network

In order to explain the possible physical restriction imposed by the network, one simple example is analyzed **in theory** (this is an example explains the essence of the packetizing and sending concept).

Analyze:

- i. **Environment:**
1-camera system
- ii. **Functions:**
 - a. *Function_Encode*(rawFrame, qualityLevel):
Used to encode the raw frame:
Parameters:
 1. rawFrame:
Captured frame, big 2Mp size picture in Figure 22).
 2. qualityLevel:
Applied to the rawFrame, to encode the frame with high quality.**Output:** encodedFrame:
Result of the Function_Encode
 - b. *Function_Network* (encodedFrameSize, payloadSize, headersSize):
Used to chop the encoded frame to blocks and append the network headers to those blocks.
Parameters
 1. encodedFrameSize:
Size of the encoded frame in number of bits.
 2. payloadSize:
1400 bytes
 3. headersSize:
62 bytes, see Figure 21.**Output:** networkPacket

When the raw frame is created it is quite a bit larger than the encoded frame, despite the variations based on the resolution of a frame. The encoded frame would be larger than the raw frame because of the addition of extra headings. Nevertheless, the frame which will be transferred to the network is not the raw frame but the encoded frame.

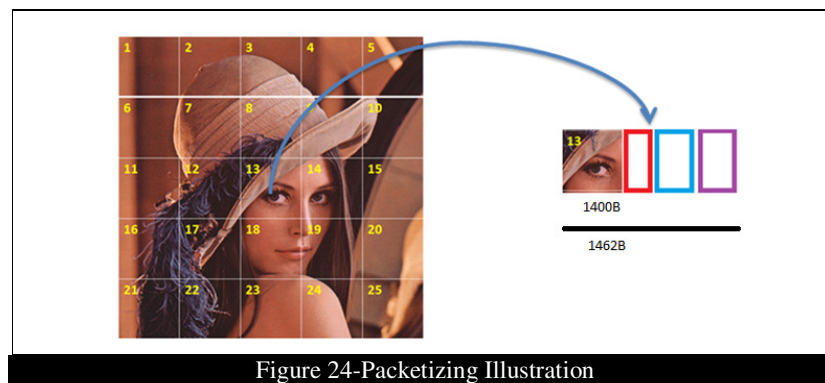


Figure 24-Packetizing Illustration

The camera captures an image at the size of 2Mp, hence the size of the raw frame is 48Mbit (2Mp; pixel is 3 bytes and byte is 8 bits, so 2x3x8Mbits).

The range of the quality level is from 0 to 100. If the quality level is 100, then the encoded frame size is almost the same as the raw frame size. Based on the information, the quality level is assumed 100 and the size of the encoded frame and the raw frame are the same: 48Mbits. In order to find the possible number of packets that are derived from the encoded frame the following equation is used.

$$\text{NumberOfPacket} = \frac{\text{encodedFrameSize}}{\text{payloadSize}}$$

Equation 1- Compute Number of Packets

$$\text{NumberOfPacket} = \frac{48\text{Mbits}}{1400 \times 8\text{bits}} = 4286 \text{ packets}$$

Additional headers to individual payload is 62 bytes (62x8bits), see Figure 21. The total size of the headers for encoded frame is 4286x62x8bits; 2125856bits.

The total size of the encoded frame which is transferred to the network is derived from the equation as shown below.

$$\begin{aligned} \text{StreamedFrameSize} &= \sum_{i=1}^{\text{NumberOf Packet}} 62\text{bytes} + \text{encodedFrameSize} \\ &= 2125856 + 48000000 \\ &= 50125856\text{bits} \end{aligned}$$

The result shows that the size of all the transferred network packets together is larger than the encoded frame.

The network switch to which the packets are transferred, has a 100Mbitpersecond capacity. This capacity shows that the switch can transfer 100.000.000bit per second. In order to find the possible frame bit rate on the network Equation 2 is used.

$$\text{FrameRate} = \frac{\text{NetworkCapacity}}{\text{packetizedFrameSize}}$$

Equation 2-Compute Frame Bit Rate

$$\text{FrameRate} = \frac{100000000\text{bps}}{50125856} < 2 \text{ frame per second}$$

The computation indicates that in one second the application on the camera can properly stream only one full frame but not two frames. The result shows that even if there is no more than one camera streaming, the size of the raw frame, the applied quality level to the raw frame, and the network capacity affect the network load too much. The created frame rate by the sensor board is 25FPS but the streamed rate becomes 1 FPS because of the capacity of the network switch.

3.2.2.2 PC Software

On the PC a general purpose OS, viz., Windows7 is installed. The OS is commercial and does not provide flexibility to the user and source code is not available. However, the user has the flexibility to install many applications for different purposes, e.g., a multi-media player for entertainment.

The multi-media player is the necessary application for completion of the distributed video processing chain, see Figure 17. Video processing has stringent timing needs, i.e., processing has to be done under time restrictions. However, a general purpose OS such as Windows 7 does not consider the timing constraints of the video application and it can be interrupted any time by other applications. When a multi-media application is interrupted the video application cannot meet the time constraints and process the frame.

There are lots of applications which are running on the PC that are competing for the use the processor. Moreover, there is no estimation whatsoever which application is processed when. The processors are fast enough but there is no prediction on the utilization of the resources.

3.3 2-Camera System

The 2-camera system is composed of two cameras and one PC, see Figure 25. In this system the main physical restriction is the bandwidth capacity which is insufficient to carry out all of the packets, see section II: *Frame packetize and send*. The details for the distributed video processing are given in section 3.1.

If the 2-camera system is considered as one complete system, the shared resource is the network that is utilized by both cameras.

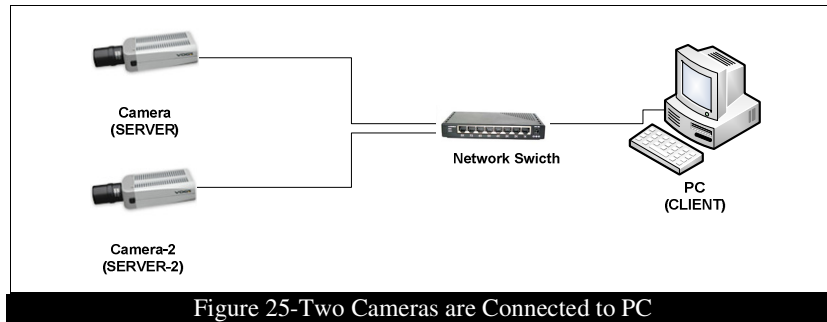


Figure 25-Two Cameras are Connected to PC

The bandwidth capacity is limited, see Table 8. Due to the characteristic of the network switch and size of the bandwidth, when two cameras are started to stream, packets are dropped by the network switch, and thus they are lost before they reach the PC. As shown in Figure 26, two cameras are streaming frames which overlap on the network switch. If the frame rate in bits is higher than the available bit rate of the switch, then packets will be dropped by the network switch.

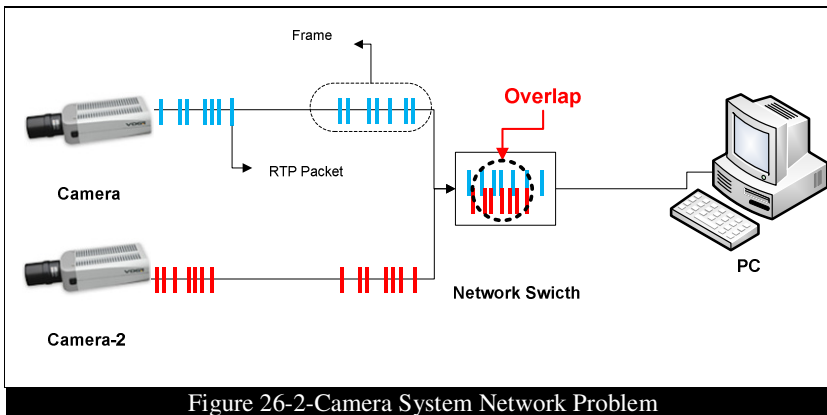


Figure 26-2-Camera System Network Problem

3.4 Summary Possible Problems

In the current project, problems are hidden within the project goals. In order to achieve the goals, problems need to be extracted. Based on this approach, the problems are divided into five sections:

Table 8-Physical Restriction System Cause Problems

#	Problem	Description	Component
---	---------	-------------	-----------

<i>1</i>	Data Loss	Frame creation process on the Sensor Platform is much faster than Frame compression and streaming on the Video Processing platform.	Camera.
<i>2</i>		Network bandwidth capacity is not enough to handle with burst transmission.	Network
<i>3</i>	High bit rate	Burst transmission	Camera
<i>4</i>	High latency	Caching of the frame	PC.
<i>5</i>	Unpredictable System	<ul style="list-style-type: none"> • System behavior is not clear. • It is unknown how much resource is utilized. 	System nodes: Camera, Network and PC.

4. System Requirements and Work Plan

This chapter presents all the functional and extra-functional requirements along with the rationale for each of them. Some of the requirements are given in the project description report. Nevertheless, new requirements are derived from the given requirements such as the choice of RTOS and multi-media player.

4.1 Functional Requirements

The Functional Requirements of this project will cover the functionalities expected from the project.

Functional requirement-1:

The System setup must contain two real-time kernels that communicate with each other.

Rationale:

The system setup contains three main components: two cameras and one PC. The video processing is started on the camera (grabbing) and finished on the PC (displaying). These separate processes are linked via the communication between the camera and the PC. The two cameras do not communicate directly with each other.

The two cameras are identical which means they have both identical hardware and software. The OS on the cameras are μ C/OS-II Real-Time Kernels. The off-the-shelf PC is pre-installed with Windows7, a general purpose OS.

In order to provide a predictable distributed application, the system components in which the application tasks are processed have to be equipped with an RTOS. For this reason, the OS on the PC has to be replaced with an RTOS. Thus, the RTOS gives a real-time task prioritization to the application over other applications running on the PC.

In general, the selection of an RTOS is subject to criteria that may vary from project to project and company to company. Table 9 shows the functional and nonfunctional requirements used for the selection of the particular RTOS used in this project.

Table 9-RTOS Functional and Nonfunctional Requirements and Rationalities

<i>Functional Requirements</i>	
FR1	Non-commercial, Linux.
FR-1 Rationale	As a project stakeholder, we cannot afford to pay for the RTOS.
FR2	Open source
FR-2 Rationale	As a developer, it should be possible to modify the RTOS code.
FR3	Intel Processor compatibility, x86-64 architecture.
FR-3 Rationale	As a physical requirement, the PC hardware architecture is x86-64.
FR4	Well documented
FR-4 Rationale	Within the restricted time of the project asking a question about the selected RTOS and waiting for the response takes some time, especially in different time zones.
<i>Non-Functional Requirements</i>	
NFR-1	Soft-real-time support.
NFR-1 Rationale	Video processing is soft-real-time application.
NFR-2	Real-time scheduler support.

NFR-2 Rationale	General scheduler does not meet real-time constraints.
NFR-3 Rationale	Software developer should familiar with the source code of the OS. In restricted time period of the project will not be enough to learn new programming language.

Functional requirement-2:

The system should support a video processing application that processes the video stream completely: from frame creation to frame display.

Rationale:

This project aims to demonstrate schedulability between different nodes: the 2 cameras and the PC. Schedulability requires scheduling within all the nodes including the network. The video processing application consists of several components and these are processed on different system nodes: on the camera (frame creation, encoding, and transfer) and on the PC (receive, decoding, and display). In order to complete the video processing application, the PC needs to be equipped with an application that can process the received frame.

The application on the PC should be a multi-media player. It is used for two aspects in connection with the camera(s) and decode the frames and display the decoded frames.

There are plenty of multi-media players, but there are some functional and nonfunctional requirements to be considered as shown in Table 10.

Table 10-Multi-Media Player Functional Requirements	
FR-1	Non-Commercial
FR-1 rationale	As project stakeholders, we do not want to pay for the multi-media player.
FR-2	Open source
FR-2 rationale	As a developer, it should be possible to modify the multi-media player code.
FR-3	Real-Time Linux Compatibility
FR-3 rationale	Not every multi-media player runs on all OSs.
FR-4	RTSP protocol support
FR-4 rationale	The given cameras can only communicate via RTSP protocol. For this reason the multi-media player must support RTSP protocol.
FR-5	MJPEG decoder support
FR-5 rationale	The frame is compressed on the camera in MJPEG format, so it should be decompressed with the same algorithm.
FR-6	Well documented
FR-6 rationale	As a software developer, without asking questions it should be sufficient read the RTOS documentation.

Functional requirement-3:

The application should be distributed but scheduled collectively.

Rationale:

The video processing application is regarded as one application. Because a frame is created and encoded on the camera, but decoded and displayed on the PC, the application is a distributed application. Scheduling for the distributed application as a whole provides the predictability to the entire system which is one of the requirements in a real-time system that needs to be met. Section 2.1 contains more about predictability.

See section 2.3.1 for a general explanation of the need for the resource reservation in order get desired quality of frame for the FR-4 to FR-8. In the system to be designed, three types of resources can be distinguished for which

reservations have to be made in a timely manner: network, processor, memory. The network is shared in the system between the cameras for streaming the data (frame or message). Processors are shared among the tasks for execution of the task inside the components. Memory is used by the task for temporal (RAM) and spatial (ROM) storage inside the components.

Functional requirement-4:

The system should have network reservation.

Functional requirement-5:

The system should have processor reservation.

Functional requirement-6:

The system should have memory reservation.

Functional requirement-7:

Automatic mode changes.

Rationale:

For the Functional Requirements 4, 5, 6, and 7

Applications running on the system nodes are competing to use the resources (processor, memory and network). Because of the limited resources during runtime the system may decide to reallocate the resources between the components, resulting in a mode change. A system mode change is an overall change in the *allocation* of resources to applications (tasks). An application mode change is a change in the *requested* resources for that application.

Functional requirement-8:

The frame quality must be adjusted automatically.

Rationale:

The capacity, the speed, and/or the size of the resources (processor, memory and network) are limited. When the frame quality is increased, it requires using more resources. One of the most critical resources is the network bandwidth: if the camera bit rate is higher than the current network bit rate, then the packets are dropped and the connection is terminated. This is an unpredictable situation that cannot be accepted in real-time systems.

Automatic adjustment is to change the quality without the user control; the quality is reduced or increased within the real-time constraints, in such a way that as the quality is increased this does not cause the connection to be terminated or as the quality is decreased this does not lead to an unacceptably low quality level (the frame content is barely discernible at quality level 30).

4.2 *Extra-Functional Requirements*

“Extra-functional requirements are known as critical success factors in traditional software engineering. Extra-functional requirements (also known as nonfunctional requirements or simply “-ilities”) are constraints regarding quality (e.g. usability, performance, security, maintainability) and economics (e.g. time, cost) of the process as well as the product components.”(28)

Extra-functional requirement-1:

The system should provide predictable resource sharing based on the assigned share of bandwidth.

Rationale:

Assigned bandwidth determines the attainable quality level of the frame. However, bandwidth is not the only resource that is utilized. Its usage should be in accordance with the other resources. For example, if the bandwidth is not sufficient to carry a high quality frame, the camera should not produce frames at high quality. Otherwise, resources are not utilized efficiently and the desired quality is still not received by the user.

Extra-functional requirement-2:

The camera system should be extendable to multiple cameras and PCs.

Rationale:

The experimental environment for the current project contains only two cameras and one PC. In order to provide a general solution, the new design should be applicable to systems that have more cameras and/or PCs.

Extra-functional requirement-3:

The system latency should be less than 100 milliseconds.

Rationale:

A frame process is started on the camera and finished on the PC. For this end-to-end processing the communication time latency should be less than 100 milliseconds. “Studies³ have shown that although observers can understand the gist of a novel pictured scene in a glimpse as short as 100 ms, the picture is likely to be quickly forgotten if another picture follows shortly.”

4.3 *Work Plan*

Based on the gathered requirements the project was structured in a series of actions. The following list shows the work plan:

- Prepare a PC with real-time Linux:
Investigate options, port and implement (preferably including HSF but not necessarily).
- Setup a system:
Connect 2 prepared cameras with this PC.
- Design an approach for distributed reservation:
Protocol: interfacing with local OS.
- Add network resource monitoring and subsequent response (mode change)
[use application simulation]
- Investigate options for extending RT Linux with reservation (servers + e.g. RELTEQ + implement).

³ Intraub, 1979, 1980; Potter, 1976; Potter & Levy, 1969; Potter, Staub, & O'Connor, 2004; Potter, Staub, Rado, & O'Connor, 2002

5. Feasibility Analysis

This chapter starts with a comprehensive investigation of the real-time operating system and the multi-media player. Then the initial measurements of the camera setup are given. Finally, the solution space is described.

5.1 Introduction

At the beginning of the project the requirements regarding the real-time operating system investigation and multi-media player selection were rather vague. For this reason, FR-1 and FR-2 were elaborated on with additional requirements. Table 9 was derived from FR-1 and Table 10 was derived from FR-2.

For the RTOS investigation, the criteria given in Table 9 are used and additional criteria are given in Appendix-A Table 32. For the multi-media player investigation Table 10 is used. After presenting the investigations in detail, the system behavior is given. The problems and the possible solutions with respect to the measurements were collected from the setup that contains the investigated software installations, see Table 33 and Figure 30.

5.2 Real-Time Linux Investigation

In order to manage real-time system events, while keeping the main hardware the same, there are many options from which to choose. Real-Time Operating Systems exhibit a wide diversity of features, and the choice of an RTOS is non-trivial.

RTOS selection criteria are typically weighted differently from project to project and company to company. However, there are some criteria, see Table 32, that are not weighted, so those are used to narrow the possible choices from hundreds of products to the one eventually installed on the PC. Based on Functional Requirement-1 and Table 9 only open-source real-time Linux implementations are considered in this project.

The selection and the investigation are based on the selected open-source real-time Linux implementation, because if the selected implementation is an extension of the Linux or the kernel patch, then the project also requires a selection of the Linux Operating System distribution.

Among all open-source real-time Linux implementation, the most popular ones are RTLinux, and RTAI. However, they are not suitable for this project for the following reasons:

- RTLinux is a hard real-time mini operating system and it is not mature enough as stated by an experienced PhD student in the SAN group.
- RTAI is a comprehensive *Real-Time Application Interface*. It is not well documented and the coding is messy as well. (29)

The number of RTOSs available for comparison is not limited to two. For this reason, Table 33 (Appendix-A) is created based on the RTOS selection criteria given in Table 32 (Appendix-A) and Table 9 and the other Linux implementations are compared. RT-Preempt Patched Linux is selected as the most suitable one.

The selected implementation is a kernel patch, so as explained in the previous paragraph the Linux Operating System distribution needs to be selected, because the patch cannot be installed directly to the PC, it has to be patched to one of the Linux distribution. In the following two paragraphs (1) preempt RT-Preempt Patched Linux and (2) the selected Linux distribution are explained.

(1) From the current stage of development of RT-Preempt Patched Linux and its support by OSADL ⁴(30) it can be expected that the RT-Preempt Patched Linux will be able to cover a majority of the typical industrial real-time demands in the near future. Besides real-time features, the RT-Preempt Patched Linux is also accompanied by the familiar Linux/POSIX programming environment, file systems, networking, and graphics. This will help to expand the application area and also facilitate the real-time programming considerably.(31) Although the current RT-Preempt source is patch-aligned with the Linux baseline, the active development of RT-Preempt (the latest release is 3.0.1- rt11, but 2.6.33-7.2 – rt30 is used in this project) and the success of lots of upstream patches indicate that it can be expected that the whole RT-Preempt Patched will be incorporated in future mainstream releases. (31)

(2) There are Linux distributions such as Fedora, Ubuntu, and RedHat. These distributions added their patches to the Vanilla Kernel, which is a pure Linux kernel. These patches come in various forms such as packages and drivers. New distributions of kernels are easy to patch with the new patches. In particular since the user does not have to compile the kernel from scratch. When the preemptive patch is patched to the Vanilla Kernel, it gives soft real-time features to the Linux Kernel.

5.2.1 Configuration of RT-Preempt Patched Linux Kernel

In order to get soft-real-time performance from the 2.6 Linux Kernel, it is enough to change its configuration to the *fully preempted kernel*. In the standard 2.6 Linux Kernel, when a user space process makes a call into the kernel (through a system call) it cannot be preempted. The configuration option **CONFIG_PREEMPT** changes this behavior of the kernel by allowing processes to be preempted if a high-priority task is ready. This feature, however, has some drawbacks. There is a slightly lower throughput and a small reduction in kernel performance, because of the added overhead of the **CONFIG_PREEMPT** option. Figure 27 shows the 2.6 Linux Kernel Layered Architecture with the RT-Preempt Patch. As can be seen from Figure 27, real-time processes and tasks are differentiated both in the user space and in the kernel space from non-real-time processes and tasks. (32)

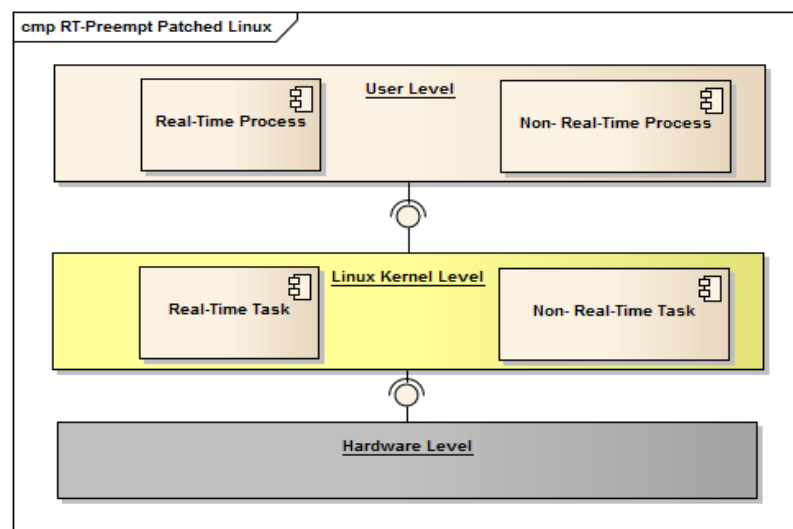


Figure 27-Standard 2.6 Linux Kernel With Preemption

⁴ OSADL is the abbreviation for the Open Source Automation Development Lab.

Finally, the Ubuntu distribution was chosen, because it is known by the project developer, and has the real-time kernel added. Installation of the RT-Preempt Patched Linux Kernel is almost the same as the installation of common Linux distributions. The installation guideline is provided on the Linux webpage. (33) After installing the RTOS, it is possible to check, by using the following command, whether the RTOS is properly installed or not.

```
$ uname -a
```

Figure 28-Command to check whether RTOS is correctly Patched or Not

5.3 *Multi-Media Player Selection (VLC)*

A multi-media player is a software package that handles media on a computer and from a network. In order to find the proper multi-media player a lot of players have been reviewed, such as SnowPlayer, KMPlayer, VLC, and QuickTime Player. Of these VideoLAN (VLC) is considered the most suitable media player for this project. It meets all the requirements mentioned in Table 10. Furthermore, VLC has the best performance in the sense of CPU and RAM utilization based on the examination in (34).

VLC player offers an intuitive API and a modular architecture that supports easy addition of new codecs, container formats and transmission protocols. VLC is commonly used in Windows, Linux and Mac OSX. (35) It is processed by the kernel as a user processes as a standard a non-real-time task. However, with a priority setting it becomes a real-time task. With the command (chrt⁵) presented in Figure 29, the user can change the priority of the thread with the Process ID (PID) \$PID_OF_THE_KTHREAD to \$PRIO in a policy such as FIFO.

```
$ chrt -f -p $PRIO $PID_OF_THE_KTHREAD
```

Figure 29-General Format of Command that turns Non-Real-Time Task into Real-Time Task

For example:

```
# the process (1024) at priority 50 in FIFO scheduling (-f)
$ chrt -f -p 50 1024
```

In Linux, priority levels range from 0-140. The range 0-99 is reserved for real-time tasks and priority levels in the range 100-140 non-real-time task priority levels.

Another way to turn a non-real-time task into a real-time task is to change the source code of the task. For this, three changes have to be made to the source code (33):

1. Setting a real-time scheduling policy and priority:
 - ✓ #define MY_PRIORITY (49)
 - ✓ param.sched_priority = MY_PRIORITY;
 - ✓ sched_setscheduler(0, SCHED_FIFO, ¶m)
2. Locking memory in such that page faults caused by virtual memory will not undermine deterministic behavior:

⁵ chrt command: Set/Manipulate real-time attributes of a Linux process [69].

- ✓ #define MAX_SAFE_STACK (8*1024)//the size is an example
- 3. Pre-faulting the stack, in such that a future stack fault will not undermine deterministic behavior:
 - ✓ void stack_pEFAULT(void) {

 unsigned char dummy[MAX_SAFE_STACK];

 memset(&dummy, 0, MAX_SAFE_STACK);

 return;

 }

For more details see the Hello World Example in (36).

5.4 System Hardware and Software Organization

After the RTOS and the multi-media player investigation and selection, we can depict the system setup, see Figure 30. It shows that the camera system consists of one PC and multiple cameras connected via an Ethernet switch.

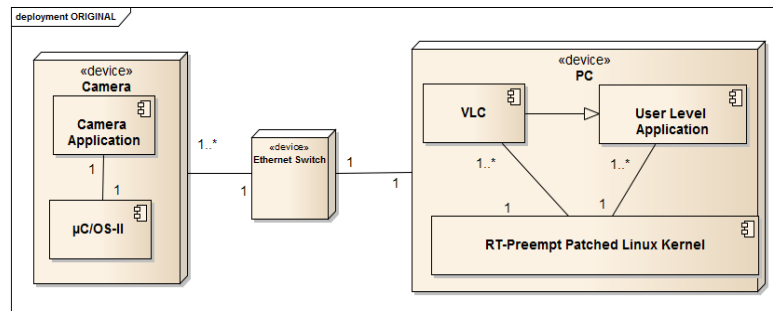


Figure 30-Camera System Software Organization

On the PC, where the RT-Preempt Patched 2.6 Linux Kernel is installed there can be multiple VLC instances and user applications as well. In the original setup there is no difference between the user level applications and VLC instances. Both are the non-real-time task. Note that RT-Preempt Patched Linux Kernel considers a task a real-time task when it is thus assigned.

On the cameras, the µC/OS-II real-time kernel and the application for video processing are installed. Initial measurements (Table 11) on this system are discussed in the next section.

5.5 Initial Experiment

The aim of the initial experiment is to show the limitations of the system. Hence, it becomes possible to figure out the exact problems and points that have to be improved. The purpose of these experiments is to show the baseline behavior of the system and determine the bottlenecks.

The initial experiments are done on the 1-camera system, see section 3.2: the software organization is as shown in Figure 30. The reason for using the 1-camera system is to have no interference on the network; the connection is only between the camera and the PC. The 1-camera system is the baseline for the 2-camera system.

In this section three main experimental considerations are explained and the reasoning of each consideration is given in Table 11. These experiments are performed from the system as presented in Figure 30.

Table 11-List of Initial Experiments and reasoning		
#	Experiment	Description - Reasoning
1	Caching (Buffering) Configuration	Caching is called buffering in VLC jargon and the cache is sized in milliseconds. It is mainly used to provide a frame smooth display. There are two consequences of buffering: (1) <i>Latency</i> : the data is stored in the buffer and is kept waiting to be processed, it is not processed on time and it causes delay. (2) <i>Data Loss</i> : if the data is kept waiting in the buffer for too long, the new data will overwrite the data inside the buffer.
2	VLC Real-Time Task	VLC is installed as one of the non-real-time user-level applications on the PC. Transforming VLC from non-real-time task to real-time task by changing its priority avoids its preemption by non-real-time tasks. It causes VLC predictable behavior.
3	Average/Worst-Case Measurements	Each task has a different execution time. Nevertheless, real-time tasks have deadline constraints that differ from the non-real-time tasks. In order to estimate the maximum execution times of the individual tasks, the worst-case execution time needs to be determined. Average case is given in order to show the system behavior difference between the guaranteed system (worst case complexity applied) and the average system (average case complexity applied). The meaning of complexity is explained in sections 5.5.3 and 5.5.4.

The following sections provide detailed information on the experiments given in Table 11.

5.5.1 Caching Optimization on VLC

The first initial experiment is to vary the default setting of the Real-time Streaming Protocol (RTSP) caching size for the received frame on the VideoLAN (VLC). The received frame is stored one second and after the buffer is filled, the frame is displayed.

In order to show the changes on the system behavior, after changing the RTSP caching size, the following two illustrative figures are given.



Figure 31-Caching Experiment: Caching Size = 1000 Milliseconds

In the original setting of VLC, the RTSP caching size is set to 1000 milliseconds. Figure 31 illustrates the caching effect on the PC screen. The camera is rotated clock-wise. It can be observed that the symbolic clock displays the time 09:13:31:43, i.e., 9 hours 13 minutes 31 second 43 milliseconds, while the PC screen displays 09:13:29:13, i.e., 9 hours, 13 minutes, 29 seconds and 13 milliseconds which was displayed on the PC screen 2 second and 30 milliseconds before being displayed on the symbolic clock.

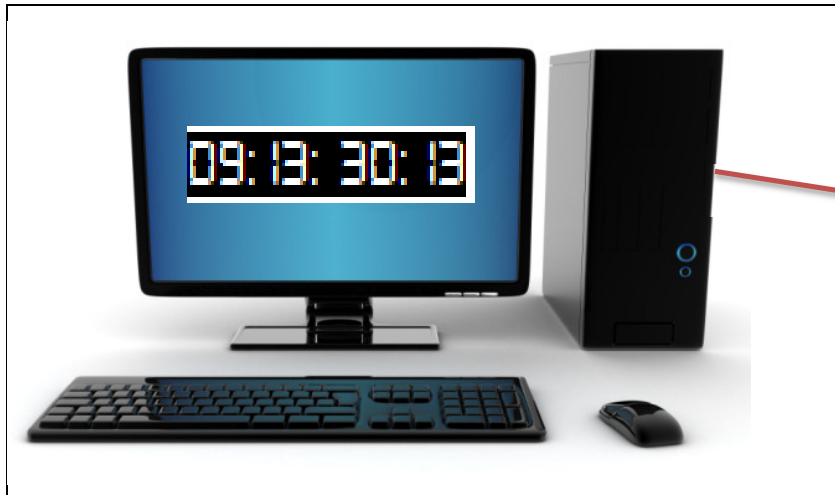


Figure 32-Caching Experiment: Caching Size = 0 Milliseconds

After setting the VLC caching size to zero, the latency is reduced to 1000 milliseconds as shown in Figure 32. The PC displays the time as 1 second and 30 milliseconds before being displayed on the symbolic clock. It can be observed that there is latency between the time on the system clock and the time displayed on the PC screen. Therefore, VLC caching is not the only reason for the latency.

By initiating a caching optimization on the VLC, the latency automatically reduces, because the VLC caches the received frame as long as the caching duration which is automatically reflected on the latency.

5.5.2 Real-Time Task VLC

The second experiment, in addition to the first one, is to change the priority of the VLC and set it as a real-time task. This will make sure that other tasks, which are non-real-time, cannot preempt VLC process.

The VLC priority was set to the kernel layer priority level 49. No data is measured in this experiment, however it is observed that there is not much difference compared to the non-real-time task behavior.

5.5.3 Average Case Complexity Measurements

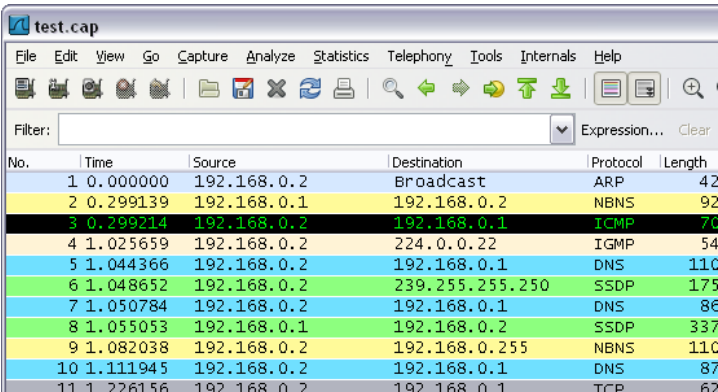
The third experiment covers the average-case complexity measurements. Average-case complexity is based on computational complexity which is the complexity of random image. The main purpose is to show the throughput rate change, based on the quality of the captured frame.

For this experiment, the camera is rotated to the stable side of the room in which there is no motion, and no defined specialty (simple or complex) for the captured frame. The only change in the environment is the light because of the windows and the sun light coming in.

Unlike the previous two experiments the quality level of frame is changed. However, the captured image and the resolution remain fixed.

In order to measure the quality level - throughput relation, one-second of video footage was recorded from this camera by using the third-party tool WireShark.(37) A one-second recording is enough to show the impact of the quality level on the frame rate.

While the camera streams frames to the PC, WireShark is sniffing the network and recording information on received network packets without differentiating between packet types, such as RTSP, HTTP, and ARP network packets. All packets on the network, either received by the PC or sent by the PC, are considered. Figure 33 shows the WireShark tool and the departure and arrival time of the packets.



No.	Time	Source	Destination	Protocol	Length
1	0.000000	192.168.0.2	Broadcast	ARP	42
2	0.299139	192.168.0.1	192.168.0.2	NBNS	92
3	0.299214	192.168.0.2	192.168.0.1	ICMP	70
4	1.025659	192.168.0.2	224.0.0.22	IGMP	54
5	1.044366	192.168.0.2	192.168.0.1	DNS	110
6	1.048652	192.168.0.2	239.255.255.250	SSDP	175
7	1.050784	192.168.0.2	192.168.0.1	DNS	86
8	1.055053	192.168.0.1	192.168.0.2	SSDP	337
9	1.082038	192.168.0.2	192.168.0.255	NBNS	110
10	1.111945	192.168.0.2	192.168.0.1	DNS	87
11	1.226156	192.168.0.2	192.168.0.1	TCP	62

Figure 33-WireShark Output Example

In order to gather the data the following three steps need to be executed:

1. *Change the Quality Level:*
 - a. User connects to the camera via the website located at <http://192.168.7.96>.
 - b. User changes the quality to the desired level.
2. *Receive frame:*
 - a. User should connect to the camera via the VLC located at <rtsp://192.168.7.96/media>.

- b. User presses the PLAY button on VLC interface.
3. *Monitor the network traffic:*
The WireShark tool should be opened. The information of all the network packets (transferred-received) is listed in the WireShark interface.

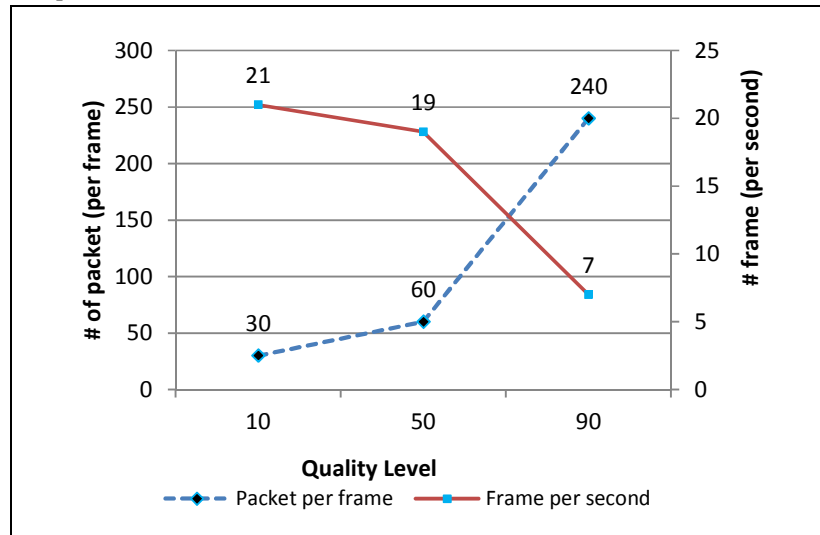


Figure 34-Frame Rate and Number of Packet per Frame of Image at Different Quality Levels for Average Case Measurements

Figure 34 shows the frame rate in relation to the changes on the quality level of the frame, from 10 to 90: 10-50-90. The figure depicts that even when the frame is the same, the quality affects the throughput noticeably as to frame rate and the number of packets for each frame. When the quality level is increased the frame rate drops. On the other hand, as shown in Figure 34, the number of packets per frame increases. In fact, the number of packets per frame increases to such an extent that also the total number of packets increases as shown in Figure 35.

The quality level difference between 10 to 50 and 50 to 90 is the same: 40. However, the increase of the number of packets per frame is more than linearly proportional to the increase in quality level. While the number of packet difference is 30 between quality level 50 and 10, the number of packet difference between quality level 90 and 50 is 180.

Along with the observation above, Figure 35 shows the total number of the packets on the network during the one-second recording as a function of the quality level. The increase in the total number of the packet is roughly proportional to the increase in the quality level.

Even as the frame rate is dropped and the number of packet per frame is increased tremendously with the increase in quality level, the packet load on the network grows almost linearly.

As a conclusion, in Figure 34 it is observed that the system behavior is non-commensurate. Figure 35 shows that the system behavior is balanced.

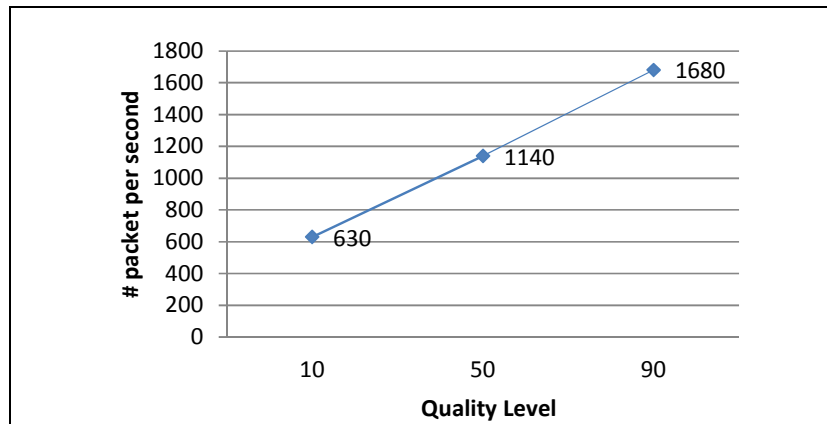


Figure 35-Total Number of Packets per Second in Different Qualities for Average

5.5.4 Worst Case Complexity Measurements

The worst-case complexity measures the resources, such as execution time, memory, bandwidth, which is required in the worst-case. It gives an upper bound on the resources that are required to be used.

Worst-Case Execution Time (WCET) is a software performance metric that is defined as the maximum length of time a task requires to complete. In order to set the deadline of a task, the WCET should be known in advance as it is used for the task scheduling.

The WCET of a task is the upper bound for the time between the task activation and the task termination. It contains all possible time consuming processes during the task processing, such as context switch, interrupts and communication. It is useful for the schedulability analysis, because it determines the amount of freedom for scheduling a task within a period. As can be seen in Figure 36, the task with a WCET of 23 time units will complete before the end of the period as long as it is started within 7 time units from the begin of the period.

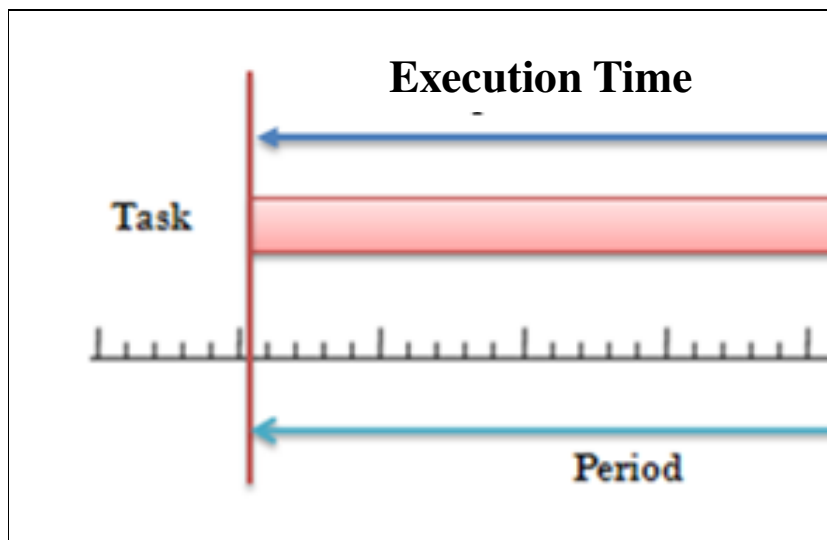


Figure 36-Task Execution Time Period in Worst-Case

There are two important inputs which affect the computation time of the encoding and packetizing-streaming tasks: the image and the quality level of the image.

The worst case image is defined with respect to the complexity of the image. The complexity needs to be at a maximum; unequal vertical and/or horizontal pixel lines, image contains x (horizontal) and y (vertical) coordinates.

If the complexity of the image is high, then the processing of the image takes a long time, i.e. much more than the average image. Figure 37 shows the worst case image which has lots of vertical strips which contributes to the complexity. For this reason, the encoding process of such an image takes more time (for more information see [38]).

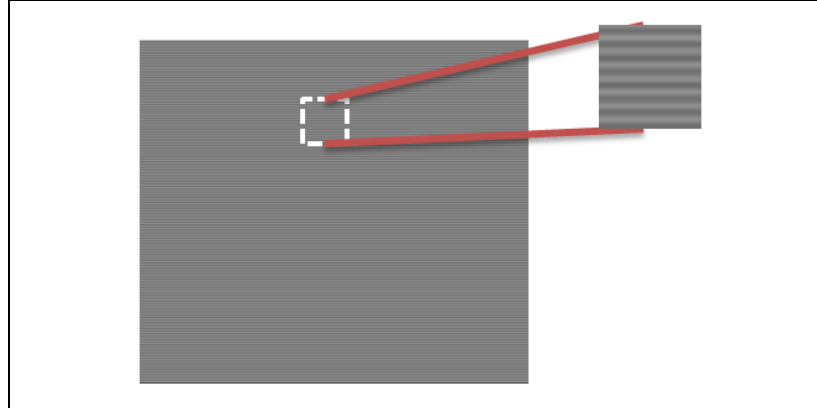


Figure 37-Worst-Case Image consisting Lots of Strips, Resolution 400x400

In order to measure the worst case execution time of the tasks within the one-camera system and demonstrate the behavior of a distributed real-time network system a test environment needs to be created. For accurate results, the test environment needs to be stable. External effects, such as fluctuating light conditions, complexity of the image, and the interruption by unknown tasks on the PC side impact the accuracy.

The test environment is composed of the ingredients as shown in Figure 38: the image (the worst case image in Figure 37) was printed out and pasted inside the box, the camera was covered with the box and it was rotated to the image that was lighted inside the box.



Figure 38-Test Case Setup Ingredients

The experiment in the test environment is the recording of the same image at different quality levels. The aim is to determine the execution time of the tasks and the throughput, while the worst case image is captured at different quality levels.

During the experiments, input parameters of the encoder function were changed by setting different quality levels, but the other settings such as resolution, the image itself, and brightness were kept fixed. After the quality level of the frame is changed, streamed frames are recorded on the PC side by using the WireShark tool. However, it is observed that on the PC side frames are not the only packets received from the network. Table 12 shows all the network traffic and the bit rate while the connection is published between camera and the PC and during the streaming. The measurements are gathered when the frame is at quality level 50. There are two parts shown in the table: the bit rate during the connection publishing and during the frame streaming

- **Connection Phase:**
Incoming/ outgoing network packets while the connection is published with the camera.
- **Streaming:**
Incoming/ outgoing network packets while the streaming process is going on.

For the bit rate measurements 100Mbps bit rate is considered.

Table 12-Message Density on Network		
During the connection: the bit rate (100Mbps)		
0.001836959		
During the one-second streaming:		
Packet Type	Size(byte)	Bit rate
UDP	60	0.0001808
RTP	854(AUDIO packet)	0.004107
RTCP	68, 70	0.00002208
JPEG	1462	0.151262

As presented in Table 12, the type and size of the packets can vary, but among all the network packets, only the JPEG packets are considered in this project. While the worst case execution time of the tasks and the throughput was examined, the software on the camera was not flashed; it is uploaded to the camera via the catapult. For more information please refer to Appendix-A section Debugging.

In order to examine the received network packets, a ten-second video footage was recorded using the WireShark tool.

In order to analyze the gathered data, the following process was followed:

1. *Change the quality level:*
The quality level settings were executed manually by re-programming the source code, because the web interface did not work; see section 7.4: Main Software Problems.
2. *Choose time interval up to 10 second:*
The received frames were recorded during ten-second periods. If it was not possible to record ten-second periods, the maximum recorded data was considered.
3. *Choose 1 second interval:*
The maximum recorded data is divided into 1 second intervals and among them, choose the one with the maximum number of recorded data.

Quality Level Measurements

In this section, the numerical relation between the frames, packets and quality levels is elaborated and the observations and the results are discussed. The purpose of this measurement is to show the relation between the quality level and the frame rate. The quality level affects the processing time of the encoding function and packetizing and throughput rate as well.

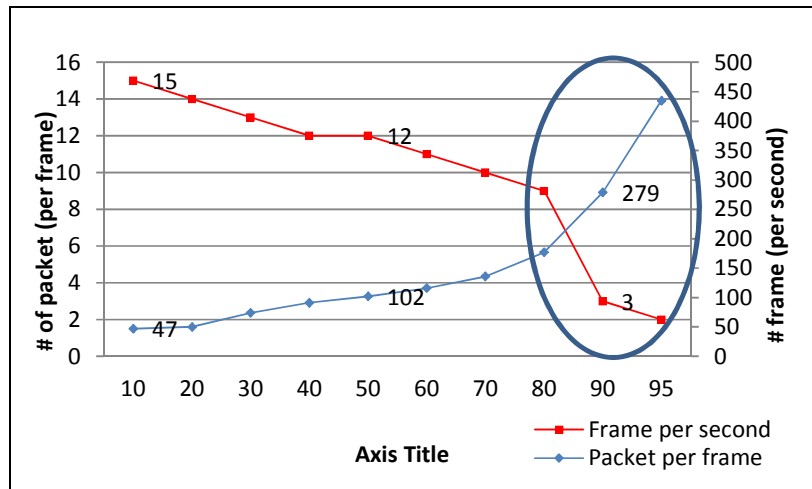


Figure 39-Frame Rate and Number of Packet per Frame of Image at Different Quality Levels for Worst-Case Complexity Measurements

Figure 39 shows the measurement, where the blue circle shows the observation which needs attention because, during the measurements the number of frame rates does not decrease significantly for the low quality levels. However, it decreases enormously when the quality level is over 80. Above quality level 90 the received number of packets is not enough to complete even 2 frames, due to the fact that the capacity of the network bandwidth is not enough. The reason is that the camera terminates the connection because the network is busy. Essentially, this result is confirmed by the analyses of the relation between the bit-rate and the quality in section 3.2.2.1 in the part on Physical Restriction on the Network.

The purpose of this measurement is to observe the system boundaries. For instance, a quality level above 80 will never be allowed for the frame to set, since the frame cannot be carried on the network because the frame size is too large at high qualities to carry it on the network.

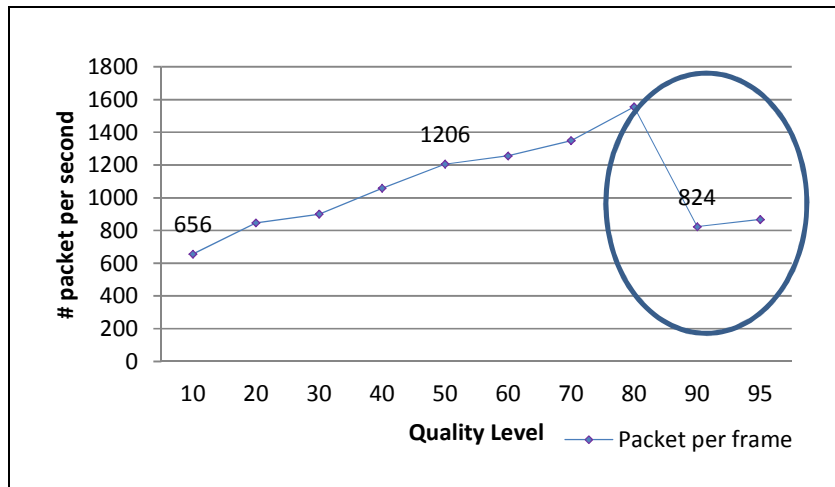


Figure 40-Total Number of Packets per Second in Different Qualities for Worst-Case Complexity Case Measurements

Figure 40 is derived from Figure 39; it shows the packet load on the network for different quality levels. It can be observed that on the PC side, the received number of packets drops tremendously.

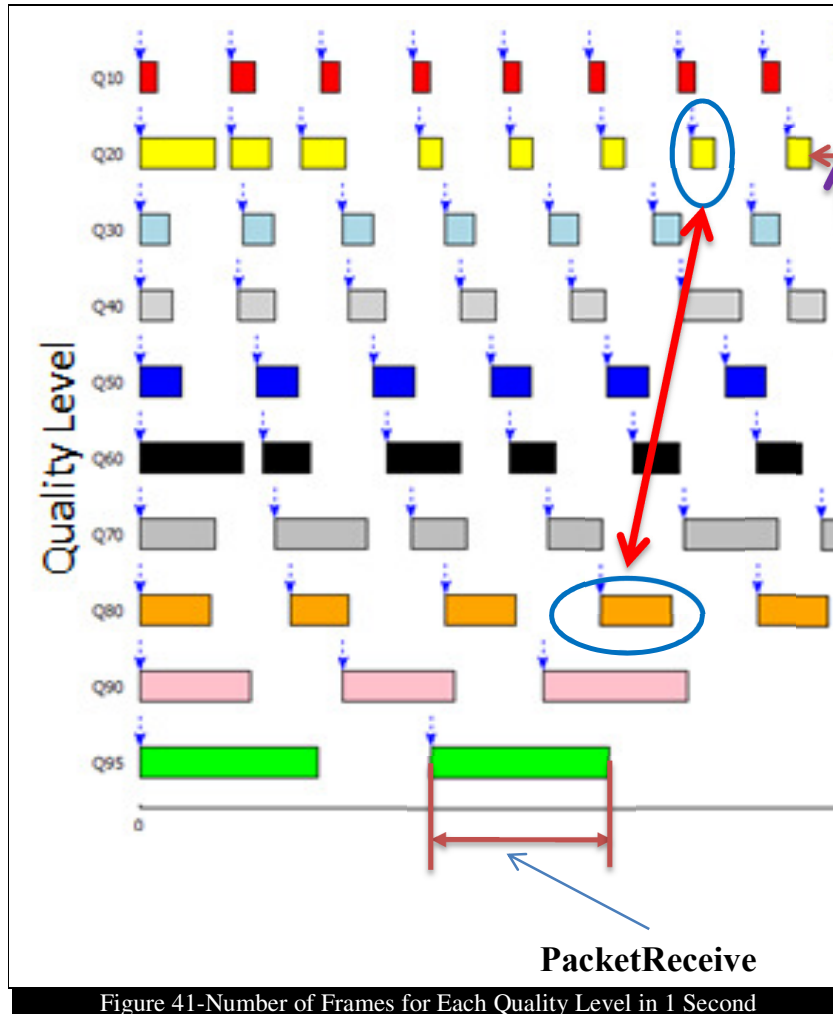


Figure 41-Number of Frames for Each Quality Level in 1 Second

In Figure 41, the y-axis presents the quality level and the x-axis presents the time. The blocks, on the horizontal line, represent the frames received by the PC: it is the measurement of the one second recording: not all the blocks at horizontal lines reach to the red vertical line, because it only considers the maximum sequential frames in one second. Each color represents the received frames at different quality levels. The received frames also present the duration of the *streaming process* and the white parts between the frames present the *encoding process* on the camera side. The *PacketReceive* label in the figure presents the time difference between the first packet received time of the frame and the last packet received time of the frame to the PC. A receive time is the time when the packet is recorded by the WireShark tool.

The purpose of Figure 41 is to visualize which quality level is proper to set when the second camera is connected to the system. As visualized in Figure 41 from top to bottom, the frame size increases due to the fact that the received number of packets increases. Conversely, the time differences between two frames stay almost the same (white parts between the two frames), regardless of the quality level.

The white parts stay almost the same, even if the quality level is low; the complexity of the image affects the execution time of encoding process. The frame size is increased due to the high quality encoded frame size. This observation helps us to decide which quality level is proper for both cameras when the second camera is connected to the system. If the frame quality level is set to 80, it is impossible to carry the frame on the network which is at quality

level 80 as well. The white space at quality level 80 shows that it is impossible to squeeze another frame which is also set at quality level 80.

The size of the encoded frame affects two things: (1) the processing time of the packetizing and sending, (2) the bit-rate on the network. Figure 42 shows the received time of the frame recorded by the WireShark tool.

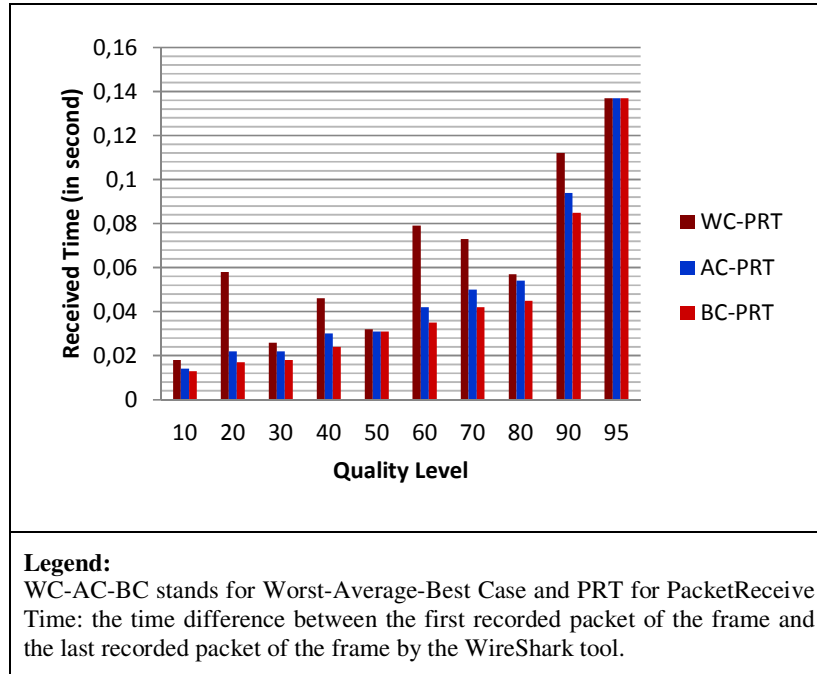


Figure 42-Time Measurements for Different Quality Level for Worst-Case Complexity Measurements (in 1 Second)

In Figure 42 from low to high, the PacketReceive time of the frames for each quality level are shown. The PacketReceive time of the frame (at different quality levels) is proportionally increased by the fact that the number of the packets increases with small amounts, except in case of a quality level above 80.

To get a better indication of the processing time in the camera we zoom in on individual frames and the individual packets within these frames: quality level 10 to 95. For this example, we select quality level 50 the 1-camera case (Figure 43) since this is also attainable in the 2-camera case.

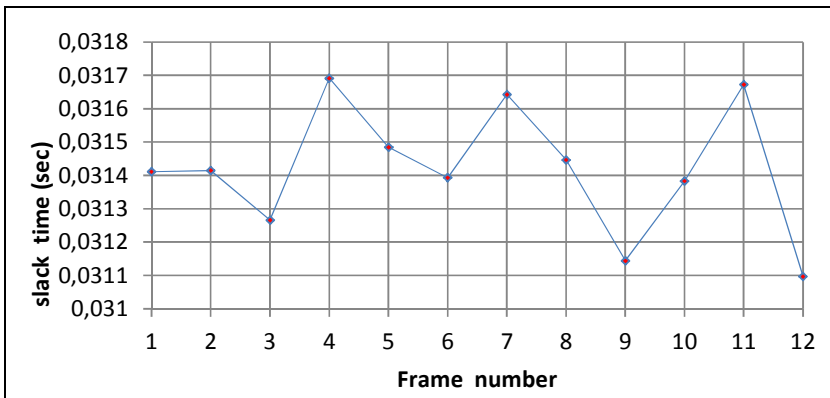


Figure 43-Time Difference between Sequential Two Frames at Quality Level 50

Figure 43 shows the time difference (*slack time*) between the first frame-last packet and second frame-first packet received times at quality level 50; see Equation 3. The arrival of each frame varies despite of the fact that all the parameters

are kept the same. These parameters include the resolution, the image, the length of the Ethernet cable, and the quality level. Note that the different length of the Ethernet cable affects the travel time of the packet from the camera to the PC.

$$SF_{i+1} = aF_{i+1} - aF_i$$

Equation 3-Slack time is Subtraction of Sequential Arrival Times; SF: Slack Time of Frame ($i+1$), aF: Arrival Time of Frame

The frames are composed of packets, so the slack time of the packets affects the received time (total receiving time) of the frame, see Equation 4. For this reason, the frames need to be examined with respect to the slack time of the packets.

$$RF_i = \sum_{j=1}^n (SP_{ij})$$

Equation 4-Total Slack Time of Packets gives Slack Time of Frame; RF: Receive time of Frame i , SP: Slack time of Packet j of Frame i

Figure 44 shows the slack time of the packets from a randomly chosen frame at quality level 50. Two conclusions can be derived from the figure: (1) the packet slack time vary, (2) two packets have the largest slack time (red circles) which affect the slack time of the frame more than the slack time of the packets in the blue circle.

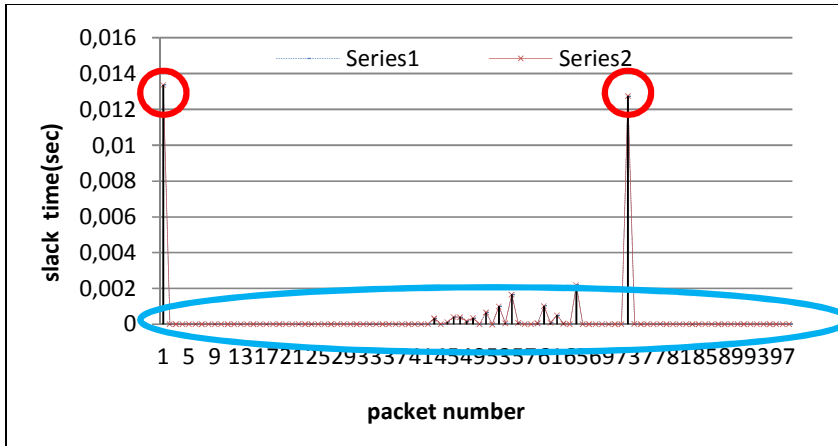


Figure 44-Arrival Time of Received Packets at Quality Level 50

There can be two reasons for the fluctuation of the slack time of the packets: (1) the packetizing and sending packet process on the camera and (2) the travel time of the packets from the camera to the PC. Nevertheless, the round-trip-time⁶ of the packet is not considered, because the length of the Ethernet cable is the same and the JPEG packet sizes are the same as well. Hence the second reason can be ignored.

⁶ Round-trip time (RTT), also called round-trip delay, is the time required for a signal pulse or packet to travel from a specific source to a specific destination and back again [71].

The first cause shows that the application on the camera is not processing *the packet sending task* deterministically. Hence, it causes a burst transmission⁷, and this in return causes data loss on the network.

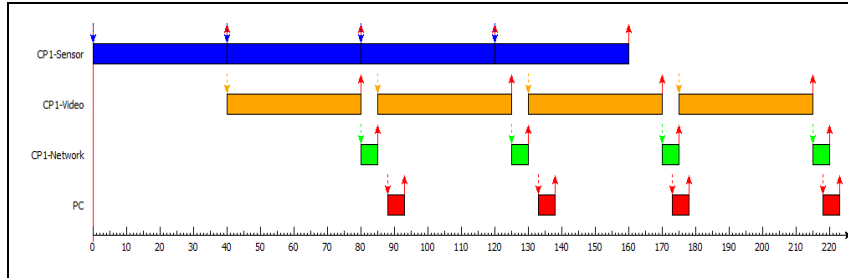


Figure 45-Processing and Transfer Time in between (Millisecond)

Figure 45 shows the symbolic representation of the time delay between the processes within the components (camera and PC). Within the camera, the frame creation process time is fixed to 40 milliseconds, and it is continuously capturing a frame, because of the hardware specifications. It is labeled as CP1_Sensor (Camera 1, Sensor Platform Task). In this task of the camera, 4 frames are created (4 blue blocks). When the frame is created, Video Task (CP1_Video) and afterwards network task (CP1_Network) is started. When the packets reach to the PC, they are processed on the PC. While the received frame is processed on the PC (red block) the other tasks continue to do their tasks. PC has to wait approximately 85 time units in order to process the first frame, however the delay between the other frames is not as much as the first frame, see the time difference from the 0 time unit to the first frame processed on the PC and time interval (delay) between the other frames on the PC (red blocks).

5.6 Problems and Solutions

Even though RTOSs are installed on the system components and tasks are real-time, the system is still not predictable. Table 13 provides the identified problems and proposes several solutions to these problems. It also indicates the feasibility of the proposed solution regarding the implementation and system restrictions.

Table 13-Problem-1 and Solutions			
Problem -1:			
<i>Data loss on the camera.</i>			
Description:			
<i>The camera is equipped to produce a frame rate of 25FPS, but it is not possible to get that frame rate from the camera. Although, Sensor board captures a frame every 40msec, the encoding and packetizing algorithm is not fast enough to process the created frame.</i>			
#	Solution	Description	Possible?
1	Increase the capacity of the resource.	Add more resources and/or increase capacity of the resources (processor and/or memory).	No, the project scope states that the existing hardware cannot be changed.
2	Enhance the encoding algorithm.	Optimize the encoding algorithm running on the camera, so that it can process the data encoding and streaming	No, we do not have the encoding source code available.

⁷ Burst transmission is any relatively high-bandwidth transmission over a short period of time. For example, a download might use 2 Mbps on average, while having "peaks" bursting up to, say, 2.4 Mbps.

		synchronously with the frame creation.	
3	Increase the network packet size.	If network packet size is increased, the number of packets will be reduced and the packetizing process will take less time.	No, since the size of the network packet is fixed it is not possible to change the size of the RTP packets.

As shown in Table 13, data loss cannot be avoided, because of the physical restrictions of the camera.

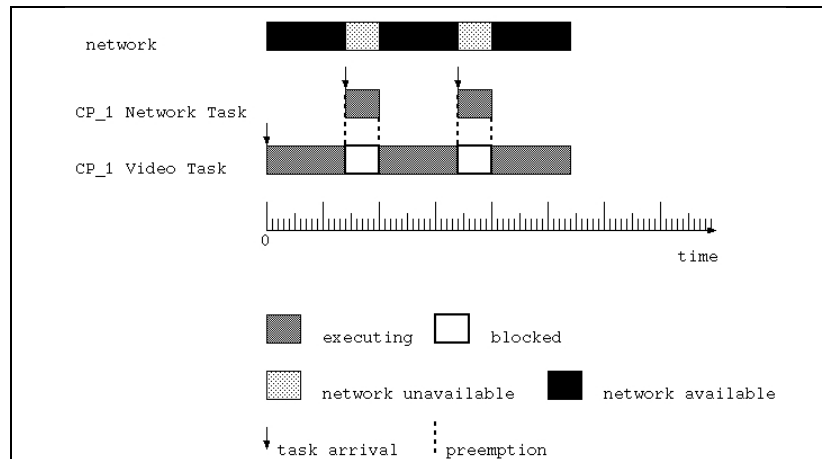


Figure 46-1-Camera System, with 1 processor the state of the processor and the network

Figure 46 depicts the 1-camera system. It shows the video task and the network task performing their respective workloads. The network tasks stream the frame through the network. When there is no streaming to the network, it is available. The network tasks are unavailable when the network task runs.

Table 14 gives Problem-2 and several solutions to Problem-2; it also gives the feasibility of the proposed solution regarding the implementation and system restrictions.

Table 14-Problem-2 and solutions			
Problem-2:			
<i>Data loss on the network (1-camera system).</i>			
Description:			
<i>High quality frames cannot be carried on the network because of the limited bandwidth capacity.</i>			
#	Solution	Description	Possible?
1	Increase the capacity of the bandwidth.	Increase the bandwidth to 100 Mbps.	No, the project scope states that the existing hardware cannot be changed.
2	Reduce the quality of the frame.	Reducing the quality of the frame will reduce the number of packets on the network.	Yes, although the system forbids specifying quality levels 0 and 100.
3	Access control on the resource (network), see 2.3.1 Resource Reservation.	Check whether the resource capacity is sufficient for the required level of the utilization.	Yes, if the initial capacity (the maximum capacity available as the network is not used), the current capacity (the capacity available at the moment in time during network utilization of the

			resource) and the required utilization are known.
--	--	--	---

The problem on the 1-camera system can be solved either by the quality adjustment or by allowing/ rejecting the connection based on the current capacity of the bandwidth. It is assumed that apart from the quality, the other parameters remain the same.

Table 15 states Problem-3 and several solutions to Problem-3; it also gives the feasibility of the proposed solution regarding the implementation and system restrictions.

Table 15-Problem-3 and Solutions			
Problem 3: <i>Data loss on the network (2-camera systems).</i>			
Description: <i>Because the camera streams the frames in a bursty manner, packets overlap and are subsequently dropped by the network, even if it is assumed that the capacity of the bandwidth is sufficient for two cameras to stream.</i>			
#	Solution	Description	Possible?
1	Increase the capacity of the bandwidth.	Increase the size of the bandwidth to 100 Mbps.	No, the project scope states that the existing hardware cannot be changed.
2	Resource Management.	Manage the utilization of the resources.	Yes, if the initial capacity, the current capacity of the resource, and the required utilization are known in advance.
3	Prevent frame overlap.	Packets are transmitted over the network and at certain instances they overlap.	Yes, if scheduling is applied to the cameras, then overlap can be avoided.

Figure 47 depicts Problem-3 where the requirement of Camera2 is 80Mbps, the network capacity is 100Mbps, but 70Mbps is already in use by the Camera1. Camera2 starts to stream 80Mbps which is above the remaining bandwidth capacity ($100 - 70 < 80$). This causes data loss on the network. In order to prevent an overload condition on the network, it is possible to control the access to the network (Problem 3- Solution#2).

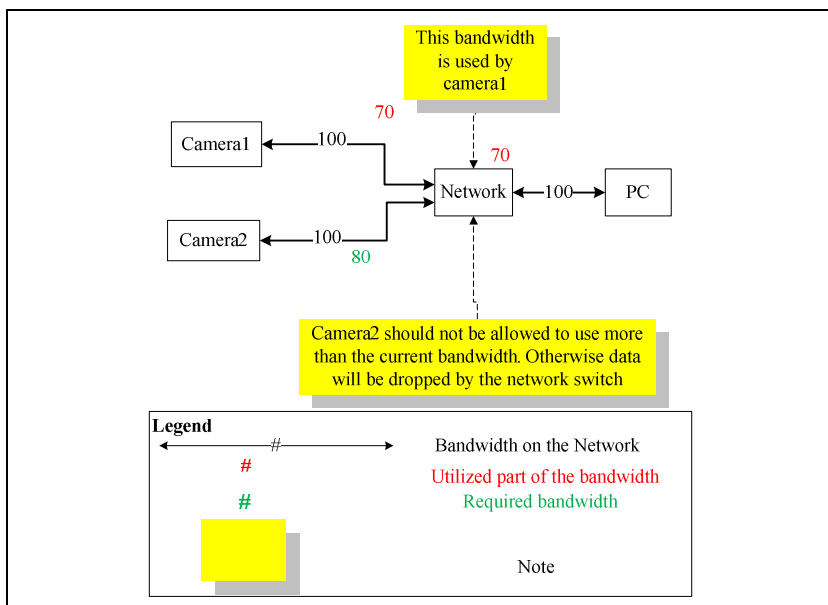


Figure 47-Problem-3: Inadequate Bandwidth

Figure 48 depicts the frame overlap: after Camera1 has encoded the frame (CP_1 Video Task), it packetizes and streams the packets (CP_1 Network Task) over the network. During the streaming period, the second camera starts to stream (CP_2 Network Task). The red block in the figure represents the overlap between the packets. If the sum of the bits streamed by the cameras exceeds the bandwidth then the packet(s) are dropped by the network switch, resulting in data loss.

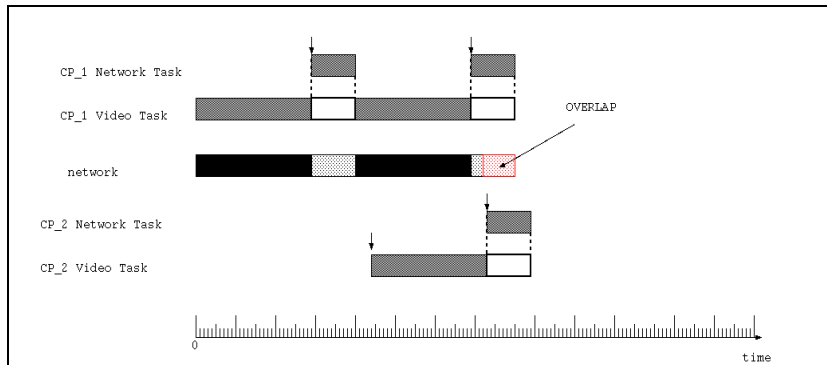


Figure 48-2-Camera System: 2 Cameras Stream Frame and Overlap Occurs, Same Legend in Figure 47

The access control to the network and prevention of the overlap between the packets are the taken decisions (Problem 3- Solution #2 and #3).

In order to prevent the overlap, the following sequence of activities has to be performed. Table 16 gives the additional aspect for Problem-3; description of the aspects and the possibility of the proposed aspects regarding the implementation and system restrictions.

Table 16-Additional Aspects for Problem_3			
#	Step	Description	Possibility
1	Scheduling	<ul style="list-style-type: none"> 1-camera system: the streaming process needs to be controlled and scheduled, because streaming causes packet overlap on the network. 2-camera system: is considered as one system therefore the cameras within the system can be scheduled like the tasks within the camera and the PC. 	Yes, in $\mu C/OS-II$ new tasks can be created, existing tasks can be re-designed, and dependencies between tasks can be synchronized with semaphores, mutexes, and message queues.
2	Synchronization	2-camera system: to apply scheduling between the cameras time synchronization method should be provided.	Yes, if it is known when the cameras stream and in which time period, then the nodes can be synchronized.
3	Enforcement	It ensures that tasks do not utilize more resources than they should. Then the system become predictable because it is clear how much bandwidth a task will maximally consume and when it will consume it.	Yes, counting the number of packets and/or defining time for the processing time of the task.

In real-time systems, the process time of the task is the most critical value to be measured. As shown in Table 18, there are three steps that need to be taken to prevent the overlap of packets and to provide a predictable system. In order to

apply the three steps, process time of the tasks should be known in advance and the deadline of the tasks should be defined.

The following section describes the given hardware and the software for the solution space.

5.7 *Solution Space*

The problems and the solutions have been given in the previous section. In order to apply the proposed solutions the environment that we work on is explained in this section. There are two main aspects: hardware and software.

5.7.1 Hardware

The hardware, see section 3.2.1, consists of the physical components:

1. 2 –Cameras,
2. 1 –PC,
3. 1 –Network Switch,
4. 1 –Catapult, see Appendix –A section Debugging

5.7.2 Software

The software description is divided into three main sections: the *Operating System level*, the *Application Level*, and finally the decision of the implementation programming language are given.

1. **Operating System level:**
 - a. Source code is written in the “C” programming language for both RT-Preempt Patched Linux and μ C/OS-II. For the development of these two OSs, C is used.
 - b. Compilers for OS:
 - i. For Linux: GNU Compiler Collection (GCC),
 - ii. For μ C/OS-II: Stretch IDE, it is targeted to the camera (embedded platform).
2. **Application level:**
 - a. **VLC player:** source code is in C, so it is developed in C
 - b. Compiler for the Application level:
 - i. In Linux; it is GNU Compiler Collection (GCC),
 - ii. In Windows, cygwin⁸(39) tool is used and it provides GCC.
3. **Implementation Decision:**
 - a. Two programming languages are proposed: C and Python. Python is very popular in the Network Programming Community, because it is easy to use and requires less lines of code (LOC), when compared with C. Threads in C are easy to manage; it compiles its code in machine language and it makes it 10 times to 100 times faster than Python. C also utilizes less memory than Python does.

The solution space provides the boundaries of the project within which the architecture is developed and improved.

⁸ Cygwin is a collection of tools which provide a Linux look and feel environment for Windows.

6. System Architecture and Design

This chapter provides a comprehensive architectural overview of the current project. It presents the architectural of the system; it defines the main components and the interaction between them by making use of several scenarios. Two diagram notations are used: a new proposal for architectural knowledge management (5) and UML.

6.1 Introduction

According to the problem analyses given in *Chapter 3*, the requirements defined in *Chapter 4*, and the feasibility study presented in *Chapter 5*, it was decided to use the RT-Preempt Linux Kernel with the Ubuntu distribution and VLC media player as fundamental components. The result of the worst-case analysis is to define the constraints of the current project. The decisions taken in Feasibility Analysis are strongly supported by the analysis with the constructed test bed. The amount of the details considered was limited by time available for the project.

First of all an overview of the distributed system is given, then the system is described in detail. Thereafter, the proposed components are explained, and the relationships between these components are given. Finally, deployment of the components is given at the end of the chapter.

6.2 System Architectural

The overall system Architecture of the current project is shown in Figure 49.

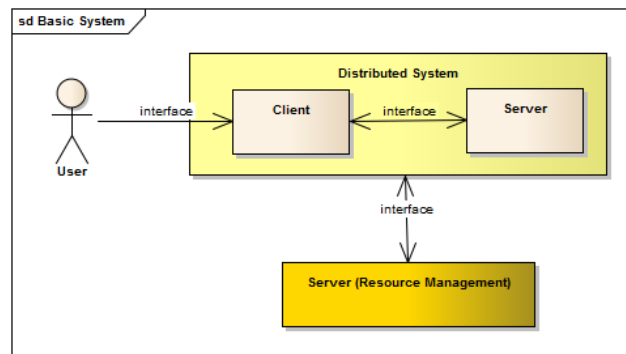


Figure 49-Overall System Architecture

- *Client (Distributed System):*
The PC, it must connect to the Server and it is responsible for processing the frames and displaying them to the user.
- *Server (Distributed System):*
The Camera, it is responsible for acceptance of the connection from the client and stream the frames to the client.
- *Server (Resource Management):*
It is the component that controls the distributed system and is responsible for managing all the resources which comprise the distributed system, such as bandwidth, memory, and processor.

Architectural Decision

Systems are seldom built following a single style and often combine a number of styles leading to hybrid designs. Two main architectural styles are applied to the system: client-server architecture and layered architecture.

Decision:

The client-server architecture style is applied.

Rationale:

The client-server architecture style is a good fit for message passing. Decoupling between the components provides maintainability and modifiability.

Figure 49 represents the high level architecture of the client-server. In the system the Client is the only entry point for the User. Resource Management is the main controller of resources of the entire system. It takes responsibility for the connection establishment between the client and the server and detects unexpected situations such as use of more resources or streaming more packets than allowed.

Decision:

The layered architecture style is used for the system.

Rationale:

The layered architecture facilitates the separation of concerns between the components. This separation of concern makes the system robust and easier to develop and maintain.

Figure 50 shows the Distributed System and the Resource Management Component. Within the Distributed System, there are two main nodes: the PC, and the camera. The camera and the PC have three layers from top to bottom: Application-Kernel-Hardware Layers. Resource Management Component has three layers, unlike the distributed system nodes layers, from top to bottom: Decision Maker Layer, Communication Layer and Storage Layer.

The network interface between the nodes is TCP/IP network interface.

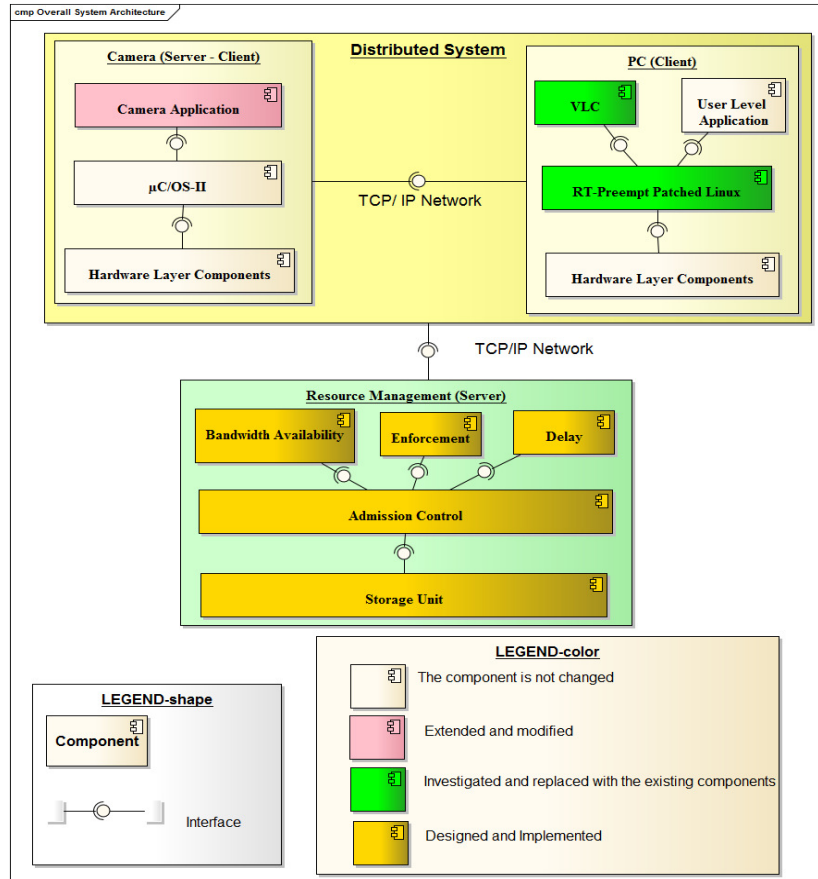


Figure 50-System Layer Architecture Component View

Within the Resource Management: the components on the *Decision Maker Layer* (Bandwidth Availability, Enforcement, and Delay) evaluates the resource utilization and synchronization before a resource is used; the component on the *Communication Layer* (Admission Control, Management Protocol) is the interface between the Decision Maker components and the other nodes (camera - PC), and the *Storage Layer* (Storage) is responsible for maintaining the values which are used by the components on the Decision Maker Layer.

6.3 System Description

Figure 51 shows communication between the Resource Management component and the original system nodes. Within the distributed system, the PC communicates with the cameras using the RTSP protocol, and streaming is performed using the RTP protocol. The Resource Management is a server and communicates with the system components with a protocol, henceforward called the ManagementProtocol that is specifically designed for the current project.

Recall from section 2.4 Real-Time Streaming Protocols, RTSP runs on top of TCP, because its responsibility is to establish the connection. RTP runs on top of UDP, because RTP does not exclusively guarantee the real-time delivery of data. ManagementProtocol runs on top of TCP, because in order to obtain coherent behavior between the reserved resource and the current resource, the given decision should be connection oriented.

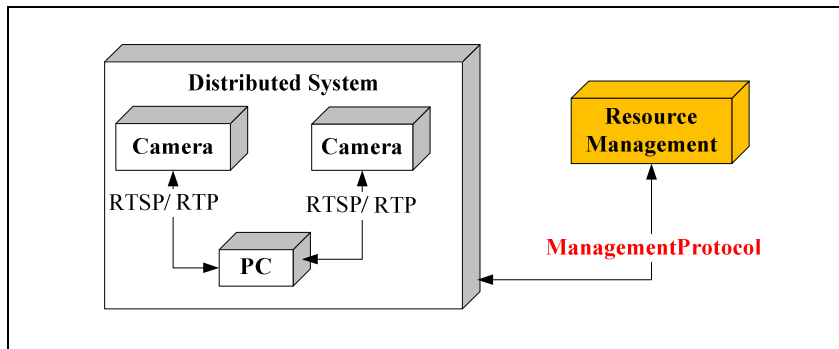


Figure 51-Communication between the system nodes

In order to control the system, the Resource Management unit typically demands commands such as frame quality information, and resource availability from the Distributed System.

The following section gives the architectural reasoning for the components of the Resource Management and later on the components is described within the Resource Management unit.

Architectural Reasoning

In the Feasibility Analysis of the system decisions have been made that lead to the introduction of a number of components and the protocol, see section 5.6. In Figure 52 these decisions are documented by means of an architectural reasoning diagram.

Architectural reasoning diagrams have three layers. From bottom to top these layers are: the *Requirements* layer; the *Decision* layer and the *Logical System Entities* layer. For each decision incoming arrows indicate the requirements (and other decisions) that gives rise to that decision and outgoing arrows indicate the entities that allow from that decision.

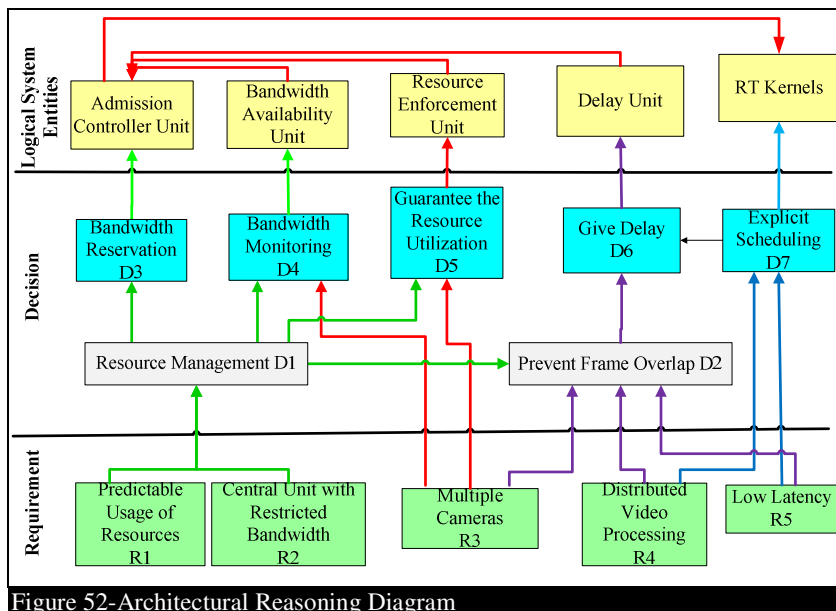


Figure 52-Architectural Reasoning Diagram

Depending on Figure 52 each block is explained as follows:

The decision Resource Management (D1) is based on two requirements: Predictable Usage of Resources (R1) and Central Unit with restricted Bandwidth (R2). Decision D1 is detailed into Bandwidth Reservation (D3) and Bandwidth

Monitoring (D4) which lead to Admission Control Unit and Bandwidth Availability Unit respectively.

The decision Prevent Frame Overlap (D2) is based on three requirements: Multiple Cameras (R3), Distributed Video Processing (R4) and Low Latency (R5). Decision D2 depends on decision D1 and Give Delay (D6) is a particular choice to affect overlap prevention. Decision D6 depends on Explicit Scheduling (D7) and decision D6 leads to Delay Unit.

The decision Bandwidth Monitoring (D4) is based on requirement Multiple Cameras (R3) and it is decision D1 in detail which leads to Bandwidth Availability Unit.

The decision Guarantee the Resource Utilization (D5) is based on requirement Multiple Cameras (R3) and it is decision D1 in detail which leads to Resource Enforcement Unit.

The decision Explicit Scheduling (D7) is based on requirement Low Latency (R5) and Distributed Video Processing (R4) which leads to RT Kernels.

The units on the Logical System Entities Layer apart from the RT Kernels depend on the Admission Control Unit, and the Admission Control Unit has a dependency on RT Kernels, in the sense of communication (TCP/IP network protocol interface).

The following list gives a very brief introduction to the units on the Logical System Entities layer. They are discussed from left to right, starting with the Admission Control Unit which is the most important of all:

1. *Admission Control Unit:*

It is the main component of the proposed system. There are two main responsibilities of the Admission Control Unit: (1) determination: establishing whether to accept a new connection request or not, (2) configuration: adjust the system configuration in order to accept the connection request, if it is possible.

2. *Bandwidth Availability Unit:*

It checks whether the reserved bandwidth is still used or not. It is necessary to make an accurate decision while the bandwidth is checked whether it is enough for the new request or not.

3. *Resource Enforcement Unit:*

It is used to ensure that tasks do not utilize more resources than they should.

4. *Delay Unit:* It is used to schedule the task and the nodes in order to prevent packet overlap on the network.

5. *RT Kernels:*

They are used to enforce the scheduling of the tasks. RT Kernels are explained in detail see section 2.2 and 5.2. In this chapter this unit is not explained any longer.

6.4 *Logical View*

In this section the functionality of the Logical System Entities and the relations between them are explained. Admission Control is the main component that the other components build on after the Admission Control, because if Admission Control does not admit a camera for streaming and terminates the connection, then the other components are not used.

The following sections present the components in detail. The explanation is as follows: functionality, relation among the other components and design decision.

The components are explained in the following order and all of them are called Resource Management Units:

1. The Admission Control Unit.
2. The Bandwidth Availability Unit.
3. The Resource Enforcement Unit.
4. The Delay Unit.

6.4.1 Admission Control Unit

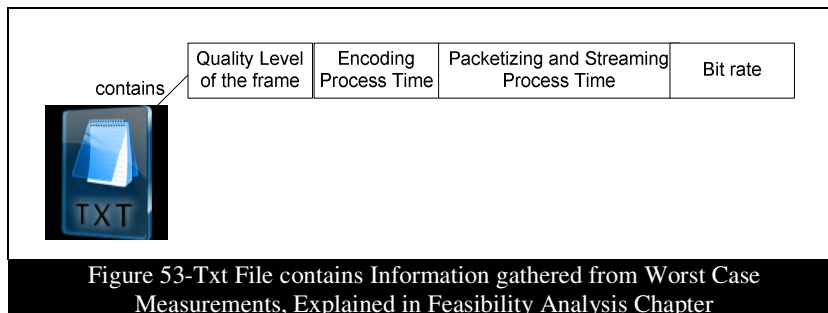
The Admission Control (AC) is created for the management of the network as a resource. The AC process decides about admittance of a new stream request. When the AC grants a new streaming request, registration and servicing phases follow.

The AC needs to understand the requested message and to provide the client with a reply that it can understand. These messages (request and reply) are parsed by the ManagementProtocol network protocol.

ManagementProtocol is proposed within the AC and adhered to by the AC. It is used for the communication between the Resource Management units and the system nodes (camera, PC). Originally, the camera application's response consisted of streaming frames but now the AC is plugged into the streaming process. ManagementProtocol is inserted to the streaming session that is provided by RTSP.

The content of the messages between the client and the AC are (1) requested frame quality, (2) the response message to the request.

The AC needs to have some criteria to give its decision to the new request. These criteria are recorded in a table (LookUpTable). It contains the data that is gathered from the worst case scenario measurements, see section 5.5.5. The content of the Look up Table is given in Figure 53.



When the evaluation is completed by the AC and the streaming request is accepted, then the camera is registered and listed by the AC. Hence, the AC knows which camera utilizes how much bandwidth. Figure 54 shows the Storage_Unit which contains the Registration List and the content of the list.

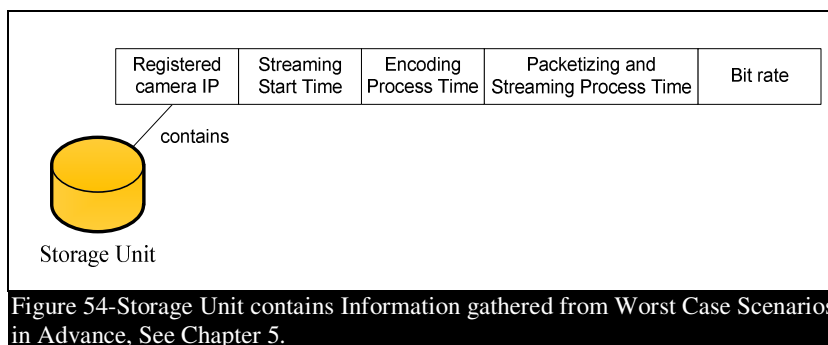


Figure 55 shows message sequence of the AC with the other components (VLC, Camera Application, LookUpTable and Registration List) via the ManagementProtocol. In this figure, the AC grants the connection. The sequence of the message is started with the User. He/she wants to connect to a specific camera in order to watch the frames on VLC. The session request is sent by VLC to the Camera Application which asks the AC to be admitted before it streams the frame to VLC. The AC evaluates the session request by comparing it with the criteria (current bandwidth and criteria listed in the LookUpTable). If the session request passes the criteria, then the AC accepts the connection and registers the camera to the Registration List.

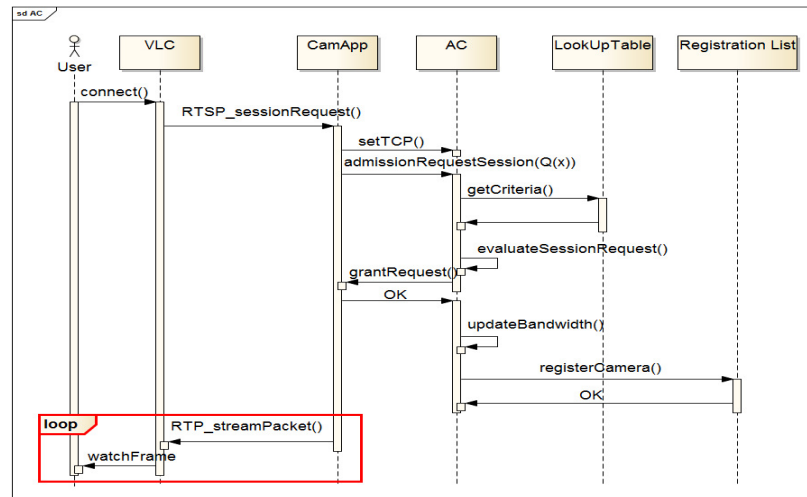


Figure 55-Message Sequence that Admission Control Unit accepts Connection from Camera Application

Figure 56 shows message sequence of the AC with the other components (VLC, CamApp (camera application), and LookUpTable). In this figure, the AC denies the connection. The sequence of the message is the same till the evaluation of the session request. The session request is denied and the AC informs the camera application about its decision. Thereafter, the camera application informs VLC that its session request is denied.

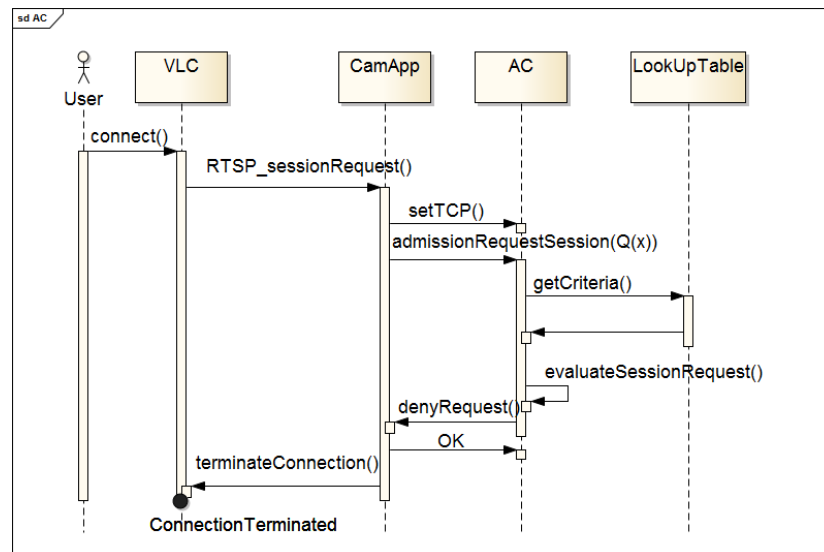


Figure 56-Admission Control Unit deny Connection

The AC gives its decision based on the scenarios given in Table 17. The table describes the scenarios and the possible reaction to the scenarios. The main idea is to grant the session request based on the availability of bandwidth.

Table 17-Admission Control Scenarios and Actions		
#	Scenario	Description
1	$BT_{req} > BT_{cur}$	If required bandwidth is higher than the current bandwidth.
	Possible Action	
	<ol style="list-style-type: none"> Reject the request: Terminate the connection, because it is not possible to provide proper bandwidth that means the capacity of the network is not enough to carry the requested frame. Adjust the quality: Reduce the quality to the available bandwidth. 	
2	$BT_{req} < BT_{cur}$	If required bandwidth is lower than the currently available bandwidth.
	Possible Action	
	<ol style="list-style-type: none"> Accept the request: The request can be accepted directly, the bandwidth is already available, but it is not known whether perceptible visual quality is good or not. Adjust the quality: Based on the information above the quality could be increased based to the current bandwidth. However, the quality should be augmented only up to average level (50). Otherwise, for the new requests there cannot be enough bandwidth. 	
3	$BT_{req} = BT_{cur}$	If required bandwidth is equal to the current bandwidth.
	Possible Action	
	<ol style="list-style-type: none"> Accept the request: The request can be accepted directly, the bandwidth is already available, but it is not known that perceptible visual quality is good or not. Adjust the quality: The request can be reduced by decreasing the quality “5” levels. If all the bandwidth has been allocated, then there will not be any bandwidth for the unexpected situations, such as extra message streaming. 	

In the default setting of the camera, the quality level is set to 50. Before, the AC is added to the system it is possible to change the quality of the frame manually via the web interface. Immediately upon receipt of a quality level change command, encoding of the current frame is abandoned and encoding of a new frame starts using the new quality level.

6.4.2 Bandwidth Availability Unit

The Bandwidth Availability (BA) unit is created to keep decisions coherent. In order to reserve bandwidth for a particular camera the current bandwidth availability should be updated. For this reason, the BA checks whether the registered camera is still connected and whether the reserved bandwidth still is in use.

Table 18 gives the possible action of the BA regarding to the response to the ping.

Table 18-Bandwidth Availability Unit Scenarios and Actions		
#	Scenario	Description
1	Ping Response = OK	Client gives response to the pinging.

	Possible Action	Nothing
2	Ping Response = time out	Client does not respond on time.
	Possible Action	Delete the client IP address and the related data from the Storage Unit-2.

Figure 57 shows the message sequence of the BA with the other components (camera and RegistrationList). In the figure $\mu C/OS-II_1$ and $\mu C/OS-II_2$ represent the cameras. In this particular diagram, the BA gets the IP address of the registered camera and pings to it. If the client does not answer within a certain time, then the registered camera is removed from the Registration List and the current bandwidth is updated. The BA continues to do this process till all items in the list have been checked.

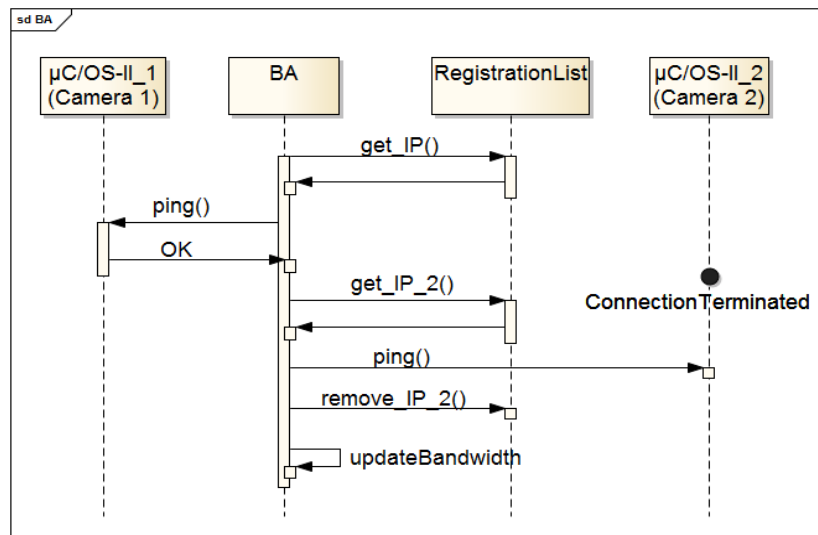


Figure 57-Connection Existence Check for Registered Cameras

6.4.3 Resource Enforcement Unit

The Resource Enforcement (RE) unit is designed primarily to enforce that system tasks do not utilize more resources than they should. The time period of the tasks and the number of packets that are streamed to the network are the most important ingredients for the RE unit, see Table 19. These two ingredients: how long process the task and how many packets streamed to the network are decided by the AC with respect to the data gathered stored in the LookUpTable based on the worst-case measurements.

Table 19-Resource Enforcement Unit Design Decision		
#	Decision	Description
1	Count packets	Camera should not stream more packets than it has been granted. The assignment is based on the worst-case scenario at certain qualities.
2	Keep time	Tasks should not utilize more resources than it has been assigned. Time period for the task should be enough to finish its task.

Figure 58 represents the enforcement on the tasks. After a connection is granted by the AC, the AC sends the information about the time period for each task within the camera application (encoding and packetizing-streaming), and number of streamed packet as well as the quality level of the frame to the camera.

The RE gives commands to the scheduler which tells it when it should start a task. Hence, each task is only executed during the period which is defined by the AC and which is controlled by the scheduler.

Before grabbing another block from the encoded frame for packetizing and subsequent sending to the PC, it should be checked whether this would bring the number of packets sent above the number allowed by the AC. Note that, tasks period and the number of streamed packet is defined by the AC based on the data gathered from the worst-case measurements.

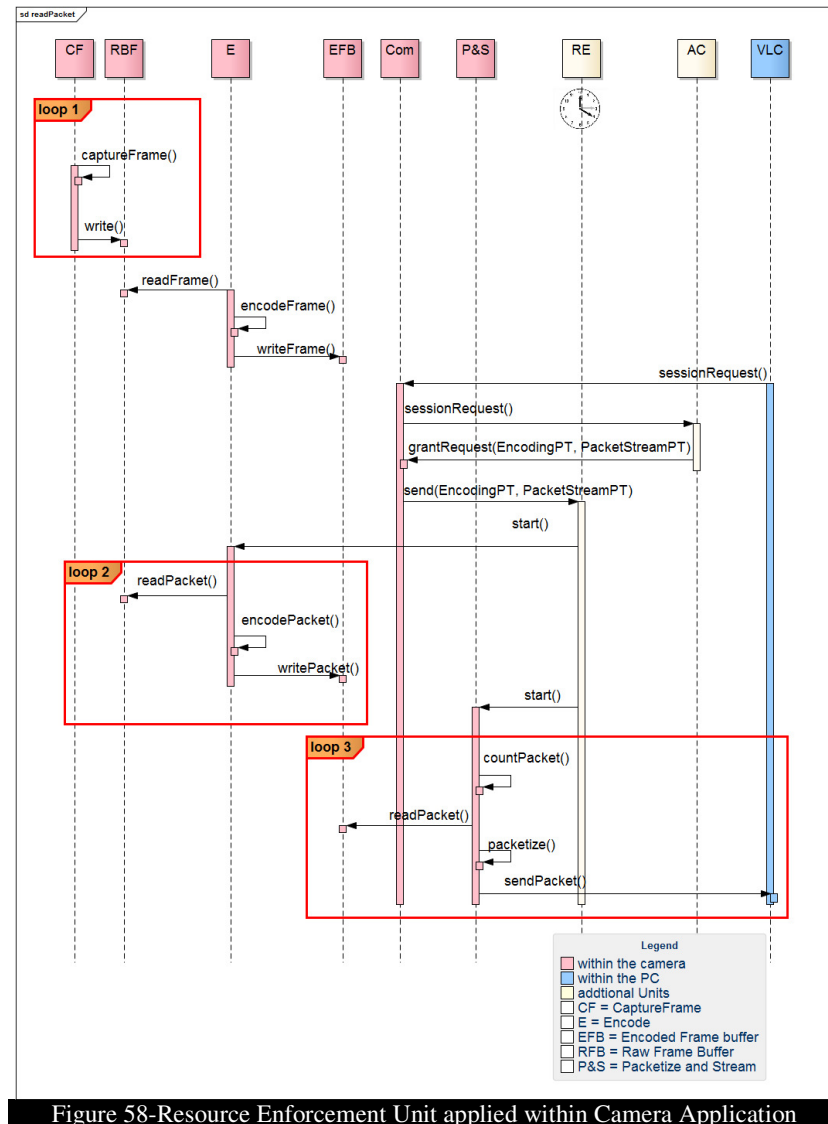


Figure 58-Resource Enforcement Unit applied within Camera Application

6.4.4 Delay Unit

The Delay unit is created primarily for the prevention of packet overlap, when the second camera connection is requested. Mainly it is an algorithm that decides how much the second camera needs to wait before starting to stream frames over the network to the PC.

Figure 59 shows the possible scenario for the packet overlap on the network when the second camera is connected. The only way is to prevent the overlap is to allow the second camera to stream while the first camera is not streaming, in the

figure blue arrow represent that the second camera streaming period filling the gap while the first camera is not streaming.

θ_1 represents the first connection start time of the streaming, E_1 represents the execution time of the streaming, and T_1 represents the period of the streaming process. The streaming process is running during the interval of θ_1 to $(\theta_1 + E_1)$ and the execution does not exceed the $(\theta_1 + T_1)$. We assume there is no jitter and drift, and the streaming process is exactly starts when the period starts.

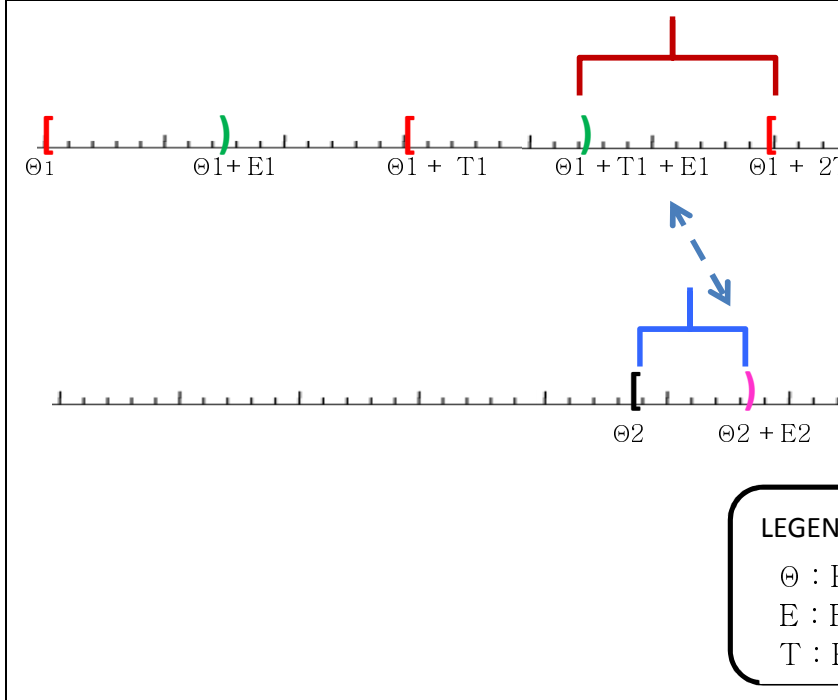


Figure 59- Possible Scenario for Preventing Packet Overlap on Network

The second camera should stream when the first camera is silent. Given periodic streaming behavior of the camera this can be achieved by a phase shift θ_2 . Discussion 1 indicates how to choose θ_2 in order to prevent overlap.

$$\exists k, (\theta_1 + kT_1 + E_1) < \theta_2 \wedge (\theta_2 + E_2) < (\theta_1 + (k + 1)T_1)$$

Discussion 1-Criteria for Increasing Phase; Item “k” is for All Positive Natural Numbers

Table 20 gives the possible action of the Delay unit, while the packets are overlapped on the network switch.

Table 20-The Delay Unit Scenarios and Actions		
#	Scenario	Description
1	2 nd camera streaming overlap	When the second camera is connected choosing the right phase for starting to stream will prevent packet overlap on the network, see Discussion 1. Relies on the assumption that packets from a camera streamed in a periodic fashion; the Resource Enforcement unit provides this.
	Possible Action	
	Provide the delay value to the camera application.	

In order to give delay to the second camera, the design decisions are given in Table 21. It contains the information who will give the decision and what the duration of the delay is.

Table 21-Prevent Overlap Design Decision

#	Decision	Description
1	Central Unit	Delay decision has to be given by a component which knows all the connections in the system.
2	Decision time	The delay decision algorithm has to be fast enough to not exceed the expiration time of the RTSP connection.
3	Enough delay	The decided delay has to be sufficient to prevent the overlap completely.

Based on the design decision in Table 21, it is needed to decide how often the overlap is prevented. For this decision two approaches are proposed in Table 22.

Table 22-Prevent Overlap Solution Approaches

#	Approach	Description
1	Delay once	After connected to the second camera, the streaming from the second camera is delayed only once.
2	Delay on a frame basis	After connected to the second camera, the streaming from the second camera is delayed every possible overlap moment.

Based on the design decisions above the last decision for the Delay unit as follows:

Design Decision : Delay once

Delay on a frame basis solution can be extension of the Start Delay solution. However, because of the limited time of the project Start Delay solution is decided to be applied. In order to give a delay, first connection information should be known.

While the second camera is not streaming, the question pops up: which task does the camera perform or not when it is not allowed to access to the network? There are two answers for that question: (1) none of the tasks are running or (2) with the exception of its streaming task the camera continues its routine processes. Table 23 gives the explanation of these two approaches and their advantages and disadvantages.

Table 23-Delay At Once, Tasks State Design Decisions

#	Approach	Description
1	Task state = Waiting	Streaming does nothing, hence it does not stream and the camera is in waiting mode, except Sensor Board part (frame creation continues). Video Task contains the encoding and the streaming functions together. If streaming has to wait it means also the encoding has to wait, because the streaming only runs whenever the encoded frame is ready.
	Pros	Cons
	No need to re-design the Video Task, the approach will not take time from the design perspective	Frame loss, while the sensor board continues to produce frames, these are not processed and the next frame overwrites the previous frame, and the frame loss will be more than the current one (3.2.2.1 Camera Software, section Physical Restriction on the Camera Application, Figure 23).
2	Task state = Running	While the streaming process does nothing, the encoding process continues.
	Pros	Cons
	Frame loss is reduced Resource (processor) on	Video Task has to be re-designed. It needs time.

the camera is not idle.	
-------------------------	--

Based on the approaches and the considering the advantages and disadvantages, the following decision is taken:

Design Decision : task state is running

The second approach is wiser than the first one with respect to the efficient resource utilization rather than leaving the system idle.

With respect to the decision (task state = running), two questions have to be answered, in order to apply the decision to the system, see Table 24.

Table 24-Video Task Division and Buffering

Question #1
Which criteria should be used to split up the Video Task How will the subtasks be synchronized? Note that the input parameter for the streaming function is provided by the encoding function.
Answer #1
The main idea of dividing the video task into two main tasks (encoding and streaming) is to continue with the encoding, while the camera does not stream to the network. In order to divide the video task: first of all, the function where the streaming starts has to be considered, and the second consideration is to synchronize the dependency between the functions. Video task has two main functions: encoding (app_video_in), and sending (STREAM_send). After encoding function, the EF information (size of the EF and location of the EF in the <i>encoded frame buffer</i>) has to be given to the sending function as an input parameter. The division is made from the streaming function and the data dependency is handled by the semaphores.
Question #2
In case the streaming process is delayed, and the encoding process is applied to a new frame, where will the newly encoded frame be stored?
Answer#2
An encoded frame is stored in the encoded frame buffer, and the size of the encoded buffer is defined when the system starts up. In the original system its size is enough for one encoded frame. In case the streaming function is in the waiting state, the encoding function will be executed and the newly encoded frame will fill the encoded frame buffer. In order to store the new encoded frame and not to lose the previous encoded frame, the buffer size of the encoded frame buffer has to be increased.

The answers are given in Table 24 is indicated in Figure 60. The figure shows the message sequence while the streaming processed is delayed by the system. In order to give delay to the streaming function, first of all the connection request should be accepted by the Admission Control Unit and the central unit should give the delay to the second camera.

Before explaining the message sequence, the items used in Figure 60 are explained in Table 25:

Table 25-Delay Unit Sequence Diagram Items

Component	Responsibility
Capture Frame	Produce a raw frame
VideoTask	Encode a raw frame
NetworkTask	Packetize an encoded frame and stream
µC/OS-II	Synchronize the tasks and provide timing
AC	Central unit send the delay information
VLC	Request connection and display the encoded frame
Communication	Accept the connection request from VLC and request connection to the AC
Buffer	Responsibility
RawFrameBuffer	Raw frame is stored
EncodedFrameBuffer_1	First encoded frame is stored

EncodedFrameBuffer_2	While the streaming function is delayed the second raw frame is stored
----------------------	--

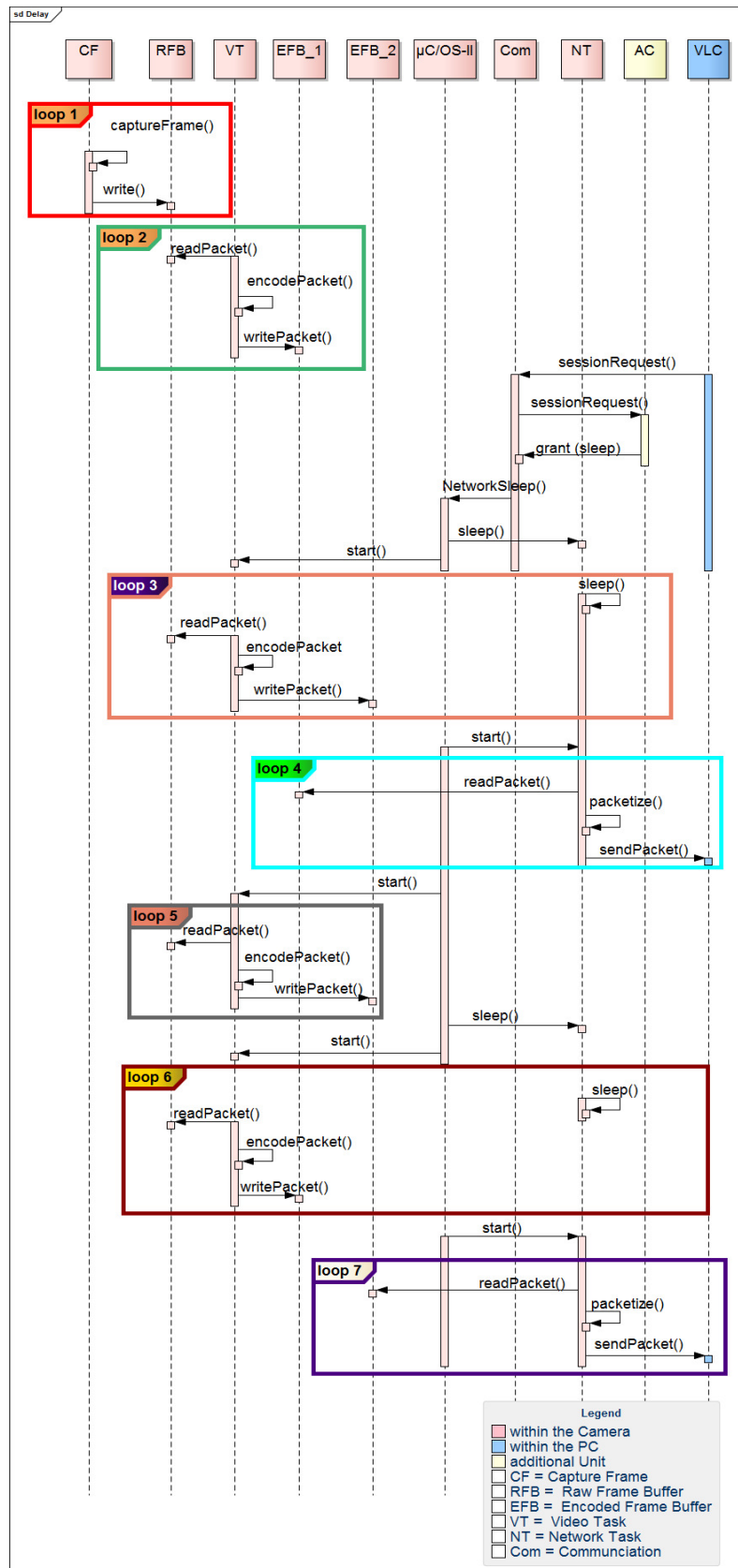


Figure 60-Giving Start Delay is processed within Camera Application

The loops within Figure 60 are explained as follows:

- *Loop_1:*
Raw frame is captured and stored in the RawFrameBuffer. This loop is processed on the Sensor Board and repeated every 40 millisecond.
- *Loop_2:*
When the raw frame is ready, Video Task gets the raw frame from the Raw Frame Buffer. Because of the limited cache size, not the entire frame but small pieces are grabbed and encoded. Each encoded packet is stored within the EncodedFrameBuffer_1. This process continues until no piece left from the raw frame.

When the connection request comes from VLC, RTSP (Communication item) steps in and creates a session request to the AC for streaming a frame. After the AC allows the camera to stream it also provides delay duration to the camera. Delay is a function that is provided by μ C/OS-II (OSTimeDelay (RTOS clock tick)).

- *Loop_3:*
When the sleep function starts for the Network Task, Video Task grabs a new raw frame from the RawFrameBuffer and encodes it till the network task wakes up and it stores the new encoded frame to the EncodedFrameBuffer_2, because the first one is already filled with the previous encoded frame. It can be observed that the sleep time cannot be larger than the streaming period of the first camera and the size of EncodedFrameBuffer is large enough to store the encoded frame at any quality level.
- *Loop_4:*
When Network Task wakes up, it starts to read the encoded frame blocks and process the packetizing and streaming until no block left inside the EncodedFrameBuffer_1. Then original routine goes and the Video Task is started again.
- *Loop_5:*
Video Task continues where it stopped to read the raw frame from the RawFrameBuffer. It encodes the rest of the raw frame and filled the EncodedFrameBuffer_2 until no block left inside the buffer.

When Video Task finished its process, Network Task needs to be started, but it has to go to sleep again and Video Task starts for the Loop_6. The main idea is to synchronize the filling buffers and depleting them. Note that the Network Task cannot be activated if the encoded frame information (encoded frame address and the size of the encoded frame) is not provided.

In order to give a delay to the second camera, the time needs to be synchronized between the central unit and the camera. If the time is not synchronized, then the delay decision will not be interpreted as same as the central unit and the camera. The following section gives brief information how the time is synchronized between the central unit and the camera.

6.4.5 Time Synchronization

This section is given in order to emphasize the importance of the time synchronization in real-time distributed systems. Based on the real-time type (hard-soft real-time) importance of the time synchronization fault-tolerance is changing as well.

In the distributed system it is difficult to synchronize the time between the nodes; this is however trivial in the centralized systems. In the camera system even

though the cameras have the same hardware and time set, it is possible that their clock times are not the same (synchronized) due to the clock drift. For this reason, the PC has to dictate the system time.

Figure 61 gives the symbolic time message passing from PC to cameras and the other way around.

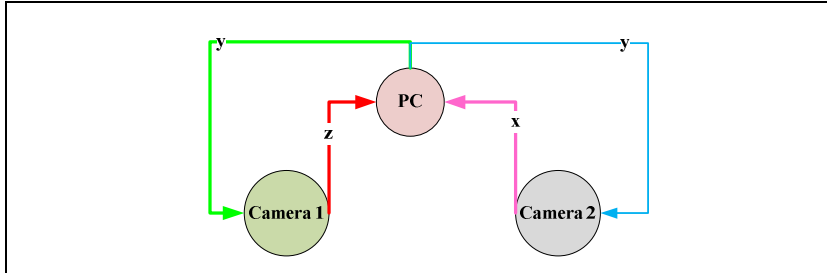


Figure 61-Time Unit Difference Representation on Cameras and on PC

In the camera system, the cameras are the same (hardware), as is the installed software on them. The only different node is the PC, regarding hardware and software. For this reason, time units on the PC and the cameras are different.

The importance of the time synchronization could be explained as follows: on the PC side the *Delay unit* takes a decision that the second camera has to wait $3y$ time units, in order to prevent packet overlap on the network. This period has to have the same meaning on the PC as on the camera side. If the camera waits more (or less) than what the PC expects, then the overlap cannot be prevented, and the Delay unit decision is taken wrongly.

In order to solve the time synchronization problem, the following has to be considered: in the camera system the nodes are interconnected by a LAN and communicate via message passing only, the system is real-time and the PC is the central unit.

Based on these considerations a lot of alternatives (protocols (NTP), algorithms (Hardware Synchronization, software synchronization) [40]) have been proposed for the time synchronization for the distributed systems. One of them is the sending a message that contains the time stamp that is stamped on the node to the message before it is streamed over the network.

The main idea is that the central unit has to inform the cameras of the correct clock time and the cameras inform the PC as well. The time unit ratio of the cameras has to be known by the PC. It can be found by comparing the speed of the processors. The speed of the processor on the PC is 3.2 GHz and on the camera is 333MHz. It means: 1 PC tick is equal to $(3.2 \cdot 10^9) / (333 \cdot 10^6)$ camera tick.

6.5 Process View

In this section, the interactions between the various components of the system are described, in particular the interactions between original components and the components added for resource management. For this purpose an Activity Diagram is used.

In Figure 62, red arrows are used to show external communication among the nodes and black arrows are used to show the internal communications within the nodes. In the figure, the most salient features are extension of the RTSP_play command and the AC unit.

The AC is triggered whenever the session request comes to the camera from VLC, because before the camera starts to stream, the streamed frame has to be controlled with respect to its quality level. RTSP protocols on the camera and the PC sending command to each other based on the RTSP protocol routine until the RTSP_play command is called by VLC. The RTSP_play command on the camera

does not send an acknowledgment message that it receives the command from VLC; it sends a request to the AC unit. The session request follows by the permission request that contains the frame quality information. The AC unit receives the frame quality information and based on the decision criteria either it is allowed the camera to stream to VLC or send a command to the camera application to terminate the connection with VLC.

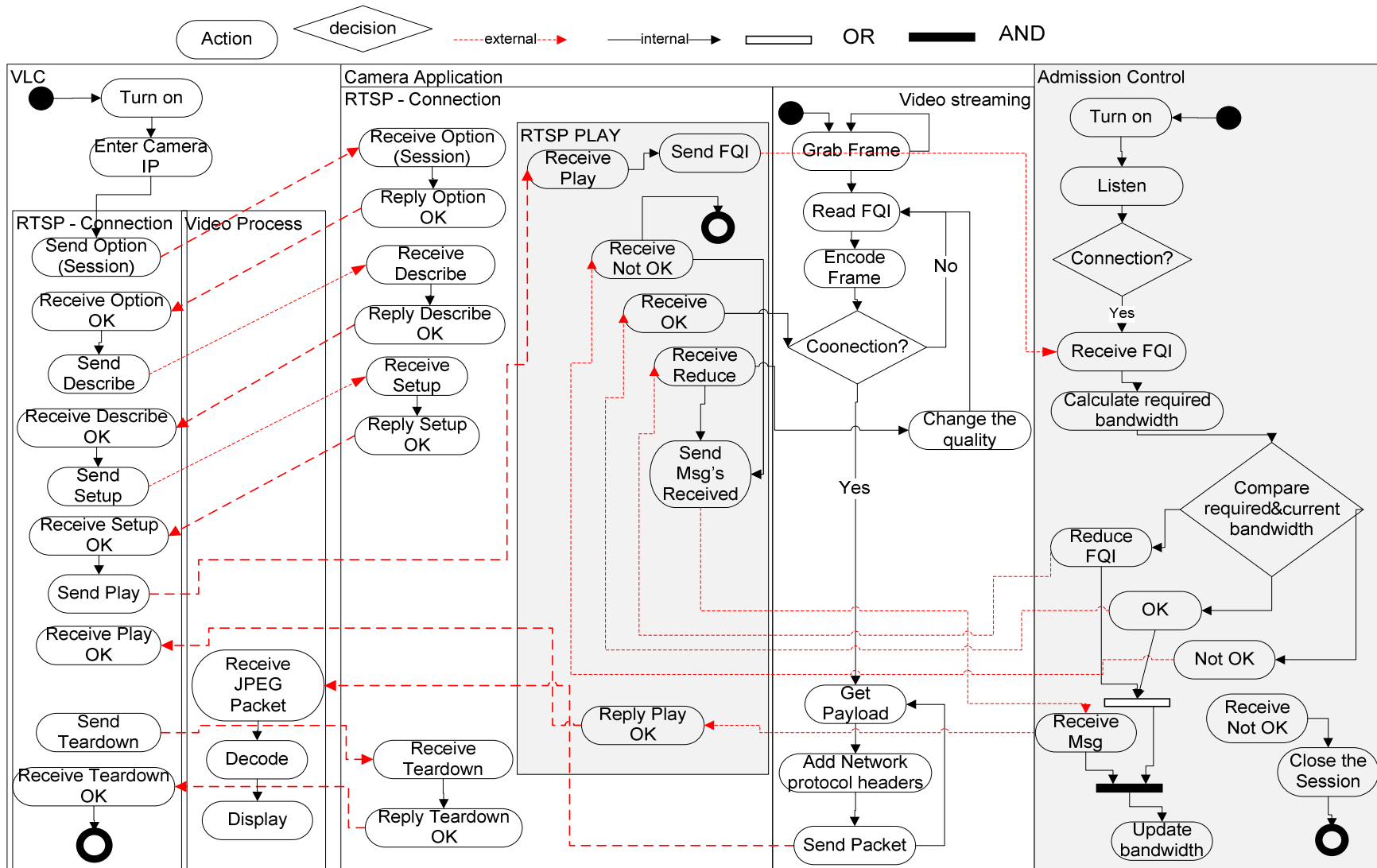


Figure 62-Interaction Overview Among VLC, Camera Application and AC

6.6 Deployment View

In this section the physical deployment of the logical system entities are explained. Figure 63 shows the deployment of the components is given together with the architectural decision diagram (5). Note that the deployment platform is constrained by the following assumption made in the original project description.

Assumptions

The camera system is composed of two cameras, one PC, and one network switch; there is no additional node.

In order to decide where to deploy the components, their features have to be evaluated based on their functionalities. Table 26 gives the components, their functionalities, and the possibilities for the deployment on the system nodes (camera and PC) along with characteristics of the nodes in the system.

There are three main rows on Table 26: Component, Hardware and Software.

- *Component:*
It is explained in order to give the characteristic of the component and find the match point with the system nodes.
- *Hardware:*
It is explained in order to decide whether the functionality of the hardware is proper for the specific component installation or not.
- *Software:*
It is examined as layers: kernel and application layer. The main idea is to focus on the implementation level whether it is possible to implement or not.

Resource Management units (Admission Control, Bandwidth Availability, Delay at once decision, Enforcement decision) are installed on the PC, because it is the central unit of the camera system. Nevertheless, they are developed as application layer software. Note that, there is communication between the nodes. Accordingly the messages coming from the components on the PC side can be interpreted by the extension of the original components on the camera side.

Table 26-Components Deployment Design Options

#	Component	Description
1	Admission Control	<ul style="list-style-type: none"> • Management Protocol: The other nodes (cameras) have to communicate with the central unit. The decision must be known by the decision maker and the related node (camera), so the protocol has to be connection oriented. • Decision Maker: Must be central decision maker.
	<i>Hardware reasoning</i>	
	The PC is the central unit of the camera system, all the cameras are connected to the PC and the PC knows all the nodes within the system.	
	<i>Software reasoning</i>	
	RTSP is the application layer protocol and the Management protocol communicates with this protocol, so they have to be at the same layer.	
2	<i>Component</i>	<i>Description</i>
	Bandwidth Availability	It has to ping the registered cameras. It must update the registration list, if one of the registered cameras is disconnected.
	<i>Hardware reasoning</i>	

	It depends where the registration list is stored, otherwise it has to communicate with the node where registration list is located, and it is time consuming and extra message load on the network. The registration List is created by the Admission Control, so it has to be installed on the PC.	
	Software reasoning	
	It is developed on the application layer. The only reason is the same place where the AC unit is developed, this is the design decision.	
3	Component	Features
	Delay at once	It has to decide how long the second camera has to wait before it starts to stream
	Hardware reasoning	
	Delay period is applied on the camera side, but the decision is send by the provided by the central unit and transferred by Management Protocol. So, the location belongs where the AC and Management Protocol is located.	
	Software reasoning	
4	It is developed on the application layer. The only reason is the same place where the AC unit is developed, this is the design decision.	
	Component	Features
	Enforcement	It gives decision about the maximum number of the packet on the network and execution time of the tasks on the camera.
	Hardware reasoning	
	Enforcement decision has to be given by the central unit, because the main idea is to schedule the entire system and not exceed the limitations of the resource utilization (processor time and the bandwidth).	
	Software reasoning	
	It is developed on the application layer. The only reason is the same place where the AC unit is developed, this is the design decision.	

Based on the given design decisions in Table 26, Table 27 gives the summary of the design decisions and the time when the components step in during the end-to-end video processing. There are three main columns: Component, Giving Decision and Application. Decision is the column explains the start of the component; Application is the column explains the reaction to the component. Application part is the supplementary of the components are given in the Decision column.

Table 27-Summary of Deployment Design Decision

Component	Decision			Application		
Admission Control	hw	layer	when	hw	layer	when
	PC	Applica tion	Before the streaming is approved on the PC.	Came ra	Applica tion	Before the streaming start.
Component	Decision			Application		
Bandwidth Availability	hw	layer	when	hw	layer	when
	PC	Applica tion	Registration list is started to fill.	PC	Applica tion	Registration list is started to fill.
Component	Decision			Application		
Giving Start Delay	hw	layer	when	hw	layer	when
	PC	Applica tion	Same time AC gave decision.	Ca mer a	Applicati on	Initial network task.
Component	Decision			Application		
Enforcement	hw	layer	when	hw	layer	when
	PC	Applica tion	Same time AC gave decision.	Came ra	kernel	Task start.

Figure 63 is much more than just a deployment view; it also shows the decision that has led to the particular deployment.

There are three layers on the diagram: requirements, decisions and the deployment of the components. In the figure: **X** represents the decision that is not taken and *cloud* represents the *idea*.

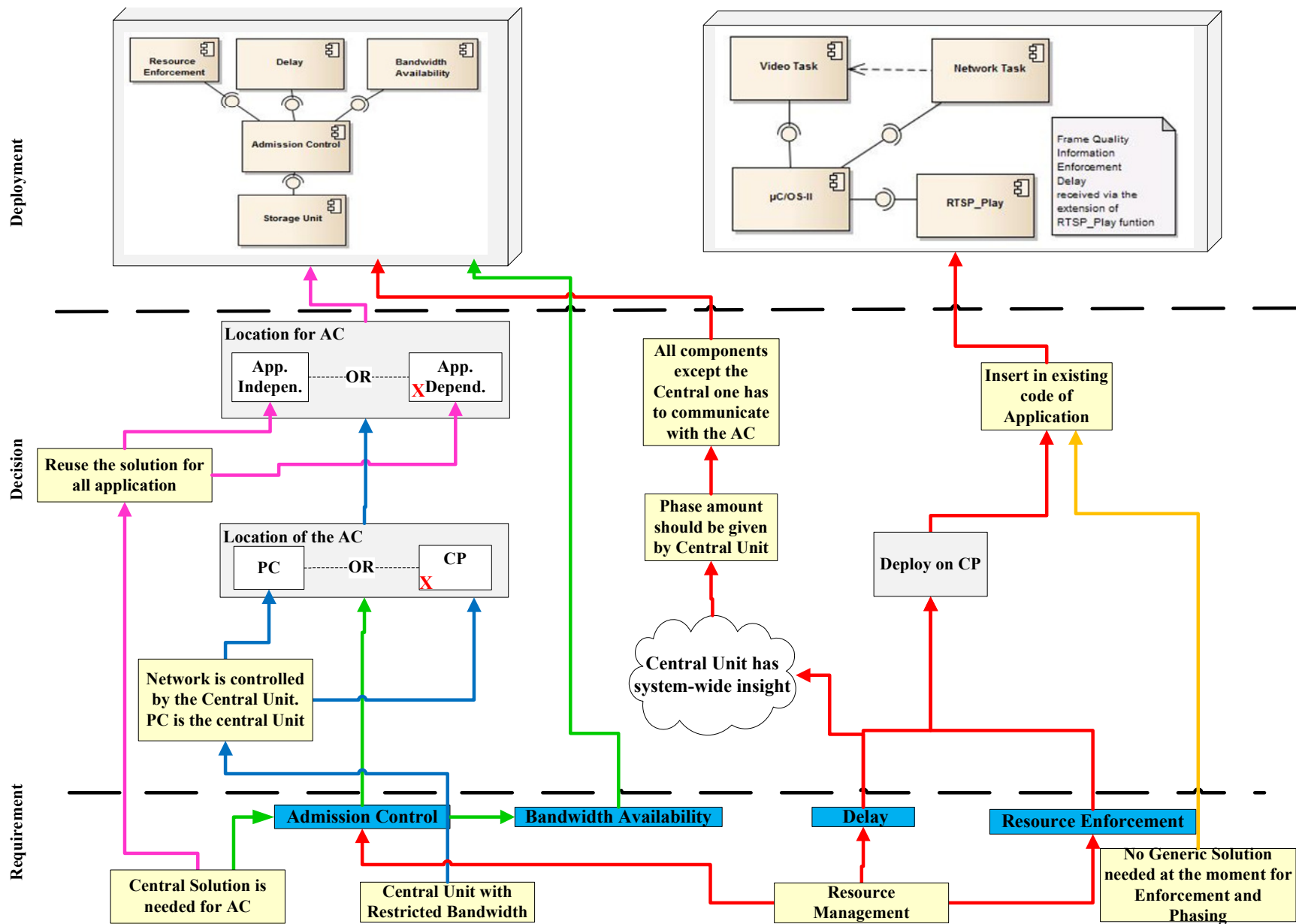


Figure 63-Components Deployment and Architectural Decision Diagram

7.Implementation and Problems

This chapter gives a detailed explanation of the component implementation. First of all, the video task division and synchronization are explained. Then a complete system description is given and the interfaces and components are explained. This is followed by a description of the Management Protocol and explanation of the delay component. The chapter is concluded by stating the software problems on the camera application and the possible resolution of these problems.

7.1 Task Division and Synchronization

In the camera application, two main applications are considered: Video Task and Real Time Streaming Protocol (RTSP) Task. The Video Task is a huge and complex task in which a lot of functions need to be executed one after another. In the original implementation, the Video Task is responsible for the encoding and, whenever a connection is established with the client via the RTSP task, for streaming.

In order to control the streaming, it is decided to divide the Video Task into two subtasks namely the Video Task and Network Task, as shown in Figure 64.

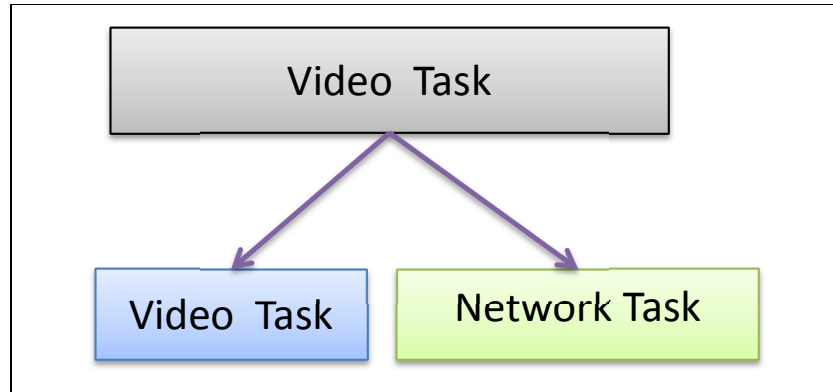


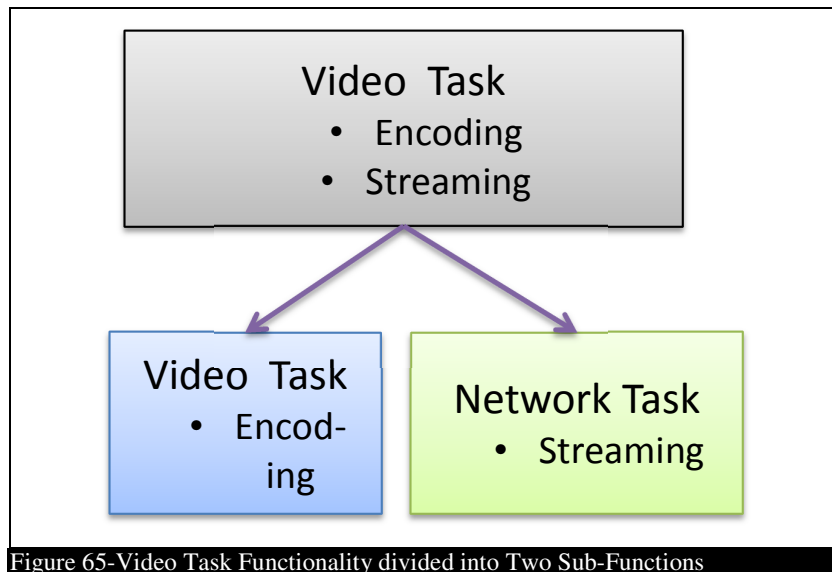
Figure 64-Video Task divided into Two Sub-Tasks

In order to divide the video task:

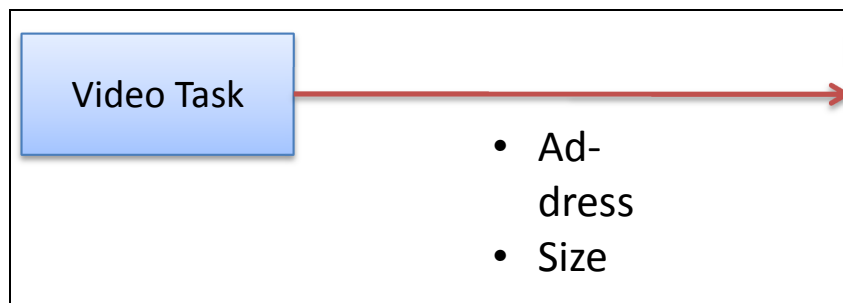
1. The Network Task has to be defined as a task.
2. The role of the Network Task has to be defined.
3. The priority level of the Network Task has to be defined.

First of all, the Network Task has to be defined as a task in the main function of the camera application.

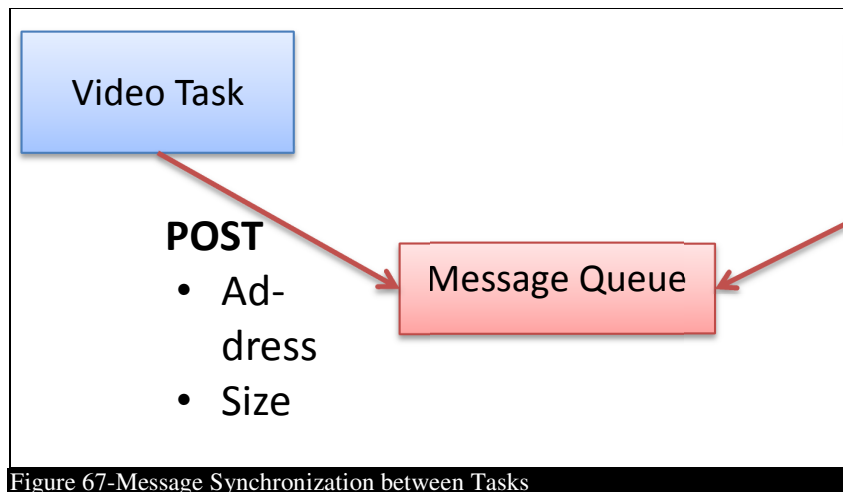
Secondly, the responsibility of the Network Task has to be defined. Recall that the original Video Task has two responsibilities: encoding and streaming; of these two the streaming functionality is given to the Network Task, as shown in Figure 65.



In order to stream the frames, the streaming function needs two input parameters: the initial address of the encoded frame in the encoded frame buffer, and the size of the encoded frame. Figure 66 shows the message dependency between these functions.



In order to decouple the tasks, the external message transfer is provided by using a message queue and the routines OSQPost () and OSQPend (), see Figure 67.



Thirdly, the priority of the Network Task has to be defined based on the responsibility of the task. The Video Task is fed by the Sensor Board which captures the new frame, and the Network Task is fed by the Video Task.

Since capturing new frames is faster than encoding frames, the input data for the Video Task is ready earlier than the Network Task input data. Based on this constraint, the Network Task has a higher priority than the Video Task. Hence, whenever the encoded frame is ready, even the Sensor Board interrupted Video Task will be executed because of the new captured frame Network Task.

Originally, the Video Task priority was set to 44. After the Video Task is divided into the Video Task and the Network Task, the task priorities are assigned as follows: Video Task Priority: 45 and The Network Task priority: 44. Note that in $\mu\text{C}/\text{OS-II}$ the priority is reduced when the priority level number is increased.

The synchronization between the Network Task and Video Task is provided by using semaphores. Figure 68 represents the task synchronization and message synchronization of the tasks (before the Video Task posts the Network Task semaphore and after the Network Task pends the semaphore).

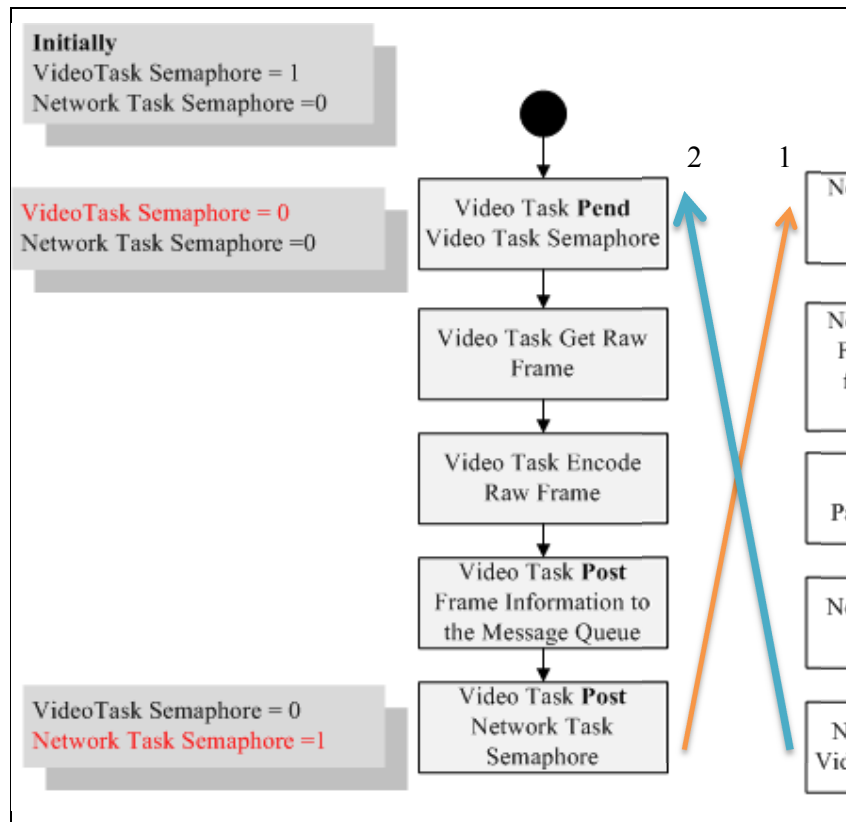


Figure 68-Task Synchronization and Message Synchronization Conceptual View

In order to start the Video Task, first set its semaphore to 1 and the Network Task semaphore to 0, because the Video Task is of lower priority than the Network Task and needs to be started first.

After the Video Task finishes the frame encoding process, it posts the encoded frame information to the message queue and posts the Network Task semaphore which becomes 1.

At the beginning of the Network Task, the semaphore is pended and the task is started. Then the Network Task pends the posted frame information of the Video Task and starts the streaming process. At the end of the streaming it posts the Video Task semaphore. Hence, the tasks will be synchronized.

7.2 System Integration and Interfaces

This section provides an explanation of the interaction of the system components. Moreover, the interfaces between the components are explained.

Figure 69 depicts the system nodes and the hardware interface situated between them.

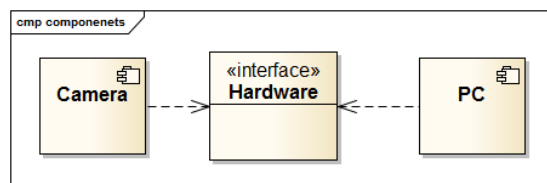


Figure 69-Interface between System Nodes

The following figures provide detailed information on the system nodes implementation.

Figure 70 shows the camera implementation. The camera application contains three main files: App_video_in, app_network and RTSP_play of which App_video_in and app_network are redesigned and RTSP_play is extended. The interaction among the components is provided via the interfaces.

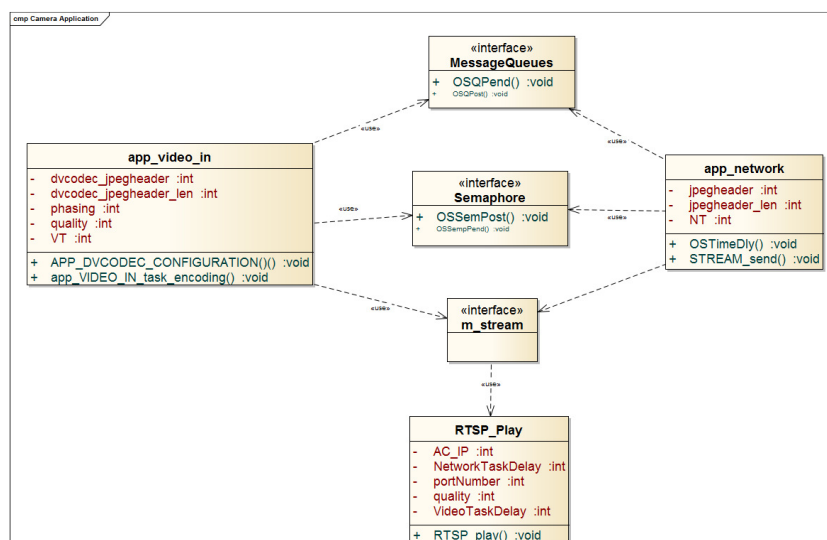


Figure 70-Implementations within Camera

Table 28 gives the content of the files within the camera application.

Table 28-Functionalities and Interfaces within Camera Application	
Functionalities	
app_video_in	This file is responsible for the encoding of the raw frame, and for storing the encoded frame information.
app_network	This file retrieves the encoded frame information and transfers it over the network.
RTSP_play	This file is responsible for performing the PLAY command which is sent by the client and extended to parse the messages which are transferred by the Admission Control.
Interfaces	
Message Queues	The Message Queues allow processes to exchange data in the form of messages.
Semaphores	Semaphores synchronize tasks.
m_stream	The m_stream contains all the real-time streaming protocol

	libraries, and is used for streaming a frame to the network.
--	--

Figure 71 shows the two main components within the PC: Admission Control and Bandwidth Availability. These two components are separated from each other. The Bandwidth Availability depends on the information stored within the address.txt file which is derived from the AC component. The address.txt file contains the information on the registered cameras.

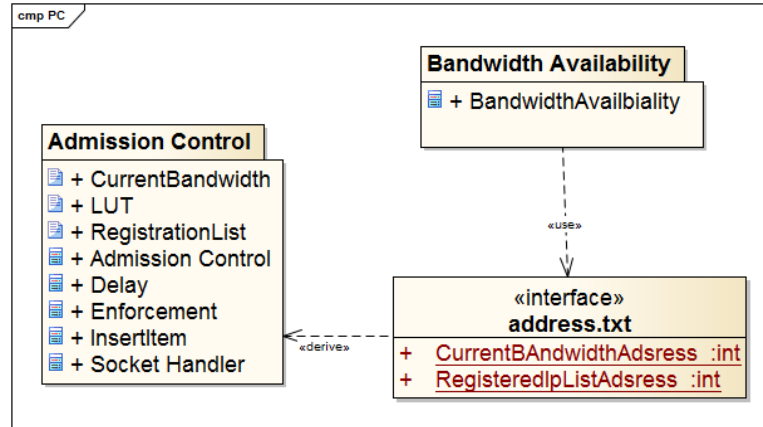


Figure 71-Admission Control and Bandwidth Availability in PC

Figure 72 presents the details of the Admission Control along with the interfaces. The artifacts (.txt files) are also given.

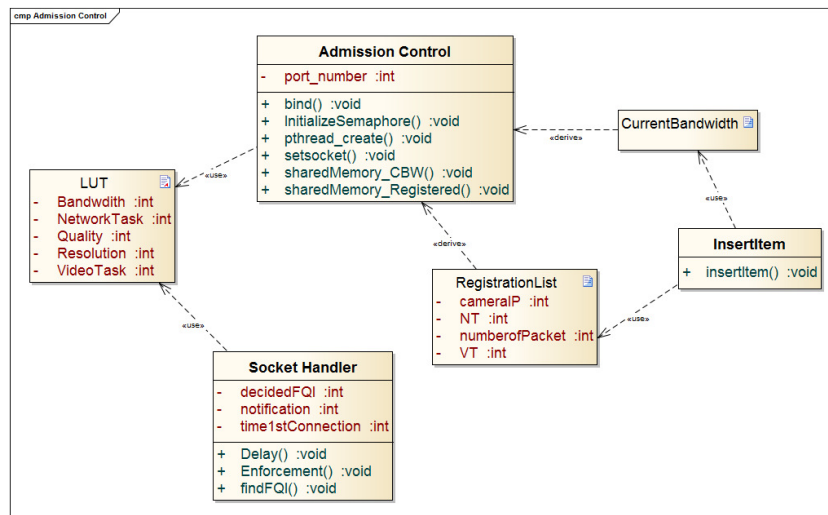


Figure 72-Admission Control

Table 29 provides the content of the files within the Admission Control.

Table 29-Functionalities within Admission Control	
LUT.txt	This file contains the information which is recorded during the Feasibility Analysis, see Figure 53.
Registration List	The Registration List is used by the Admission Control and the Bandwidth Availability units. For this reason, this list is stored in a shared memory. It is a table that contains the information of the registered camera and the processing time of the tasks (video-network).
Current Bandwidth	The Current Bandwidth is used by Admission Control and Bandwidth Availability units. For this reason, it is stored in a shared memory. It contains the current bandwidth information.
Admission Control	The Admission Control is a server and collaborates with the Socket Handler which evaluates the connection requests. It

	allows up to ten connections.
Socket Handler	<p>The Socket Handler is responsible for receiving the data in order to parse and evaluate it. It contains the functions <i>findFQI</i>, <i>delay</i> and <i>enforcement</i>:</p> <ul style="list-style-type: none"> Function <i>findFQI</i> is used to get the bandwidth utilization with respect to the received quality information from the camera application. Function <i>delay</i> is used to calculate the phasing for the second connection. Function <i>enforcement</i> gets the processing time of the tasks within the camera and number of packets for a certain quality level for the new frame.
InsertItem	This is used to update the current bandwidth. If it is updated, it records the accepted connection in the Registration List.
Bandwidth Availability	The Bandwidth Availability is used to diagnose the connection persistence. It pings the registered cameras. If a camera application does not answer within 1 second, the camera is deleted from the Registration List and the bandwidth is updated (increased by as much as what was used by the deleted camera).

Figure 72 shows the second main component called Bandwidth Availability. It only pings to the camera(s) which is (are) recorded within the Registration List. In order to ping to the camera, the ICMP protocol is used. If the pinged camera does not respond on time, the registered camera is deleted from the Registration List which is kept in shared memory.

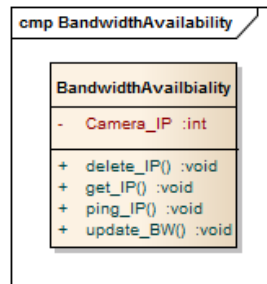


Figure 73-Bandwidth Availability

7.3 Implemented Units and Results

In this section the implemented units are described. In addition, some designed units that were not implemented are stated and the reason for this is given.

Due to the fact that the camera software was unstable, not all of the implemented components are accurately tested. The software development was performed while the camera application was in debug mode, see 5.5.4: Debugging section. Since there was only one catapult available, the units could not be tested with two cameras; while one camera was in debug mode, the other camera could not be used.

7.3.1 Admission Control Unit and Management Protocol

The original camera application functions as a server. However, during the AC connection, the camera behaves as a client and therefore the AC will function as the server. When the AC approves the connection request, the camera becomes a server again streaming frames to the PC.

There are two crucial data items that the camera application needs: (1) IP address of the AC and (2) the port number of the protocol.

While a client requests to connect to the camera, the camera application requests AC for permission. On the camera application, it is not necessary to set the IP address of the PC which has the AC deployed, because the camera application recognizes from where, VLC sends stream request. Note that if there is another server specified for the AC (not located on the PC), the IP address of the server has to be set manually to the camera application.

The camera application needs a port number. It is decided to use “7111” as a port number for the AC connection; because 7111 is out of a standard port number range.

The communication between the AC and Management Protocol on the camera application is tested using a Windows implementation (debug mode). However, the AC and the Management protocol are both developed in Linux. Even though the camera software was unstable, the system is designed based on the Linux environment.

In order to develop the AC and the ManagementProtocol, one client is implemented that communicates with the AC. Hence the parsing and the evaluating processes are tested.

Figure 74 shows the Admission Control process and the decision transfer to the implemented client (on behalf of camera) through the Management protocol.

```
Received FQI 20
Required Bandwidth 20000
RequiredBW is smaller than CurrentBW ==> AC can provide requiredBW to
the CP
AC sends the Quality : 20
AC sends the Video Task Period : 50
AC sends the Network Task Period : 7
AC will receive the confirmation form the CP that the CP receive the
notification.
CP send confirmation,so AC can add the connected CP and update the Cu
rrent BW and
Current BW before update 100000
    Before current bandwidth is updated : 100000
    Current bandwidth is updated : 80000
Current BW after update in the sharedMemory : 80000
Reserved Bandwidth 20000
Registered IP 192.168.17.141
Number of connection 1
*****
```

Figure 74-AC-Management Protocol Communication with Camera Application

Figure 75 shows the message sequence from the implemented client (on behalf of camera) to the AC.

```
AC confirms the connection
hilar@ubuntu:~/workspace/Client/src$ ./CLIENT
Client-socket() OK
Connection established...
Client will send FQI to the AC: SERVER : 192.168.17.141
CP sends FQI(20) to the AC
Waiting the 192.168.17.141 to echo back...
Echoed data from the AC : SERVER : 20
Client-read() is OK
AC confirms the connection
hilar@ubuntu:~/workspace/Client/src$ _
```

Figure 75-Camera Application Response to AC

7.3.2 Bandwidth Availability Unit

The cameras registered on the Registration List created by the AC are controlled by the Bandwidth Availability (BA) application. For this reason, the BA is also implemented in Linux. Figure 76 shows the pinging to the registered camera IPs. If the ping is not responded, then the registered camera IP is deleted from the Registration List and the current bandwidth is updated.

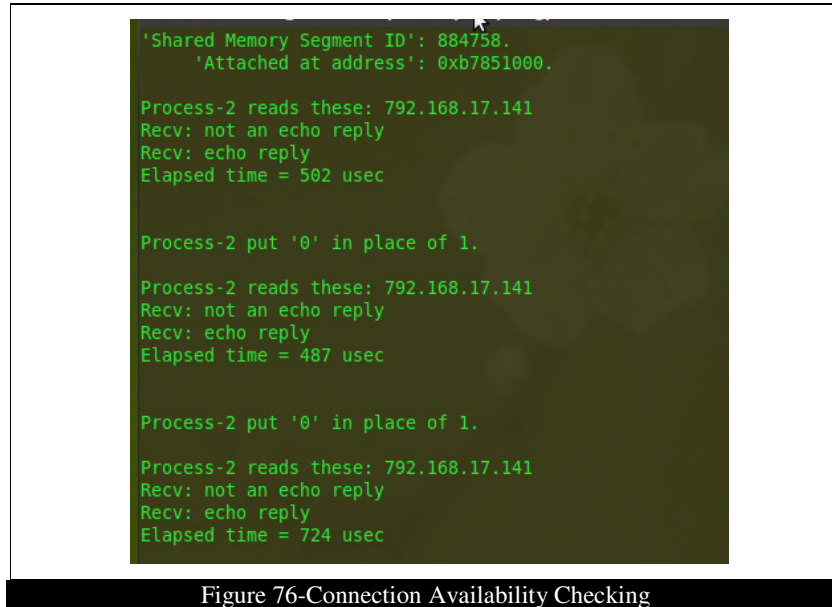


Figure 76-Connection Availability Checking

7.3.3 Resource Enforcement Unit

The Resource Enforcement (RE) Unit is designed to enforce that tasks do not utilize more resources than necessary. In addition, the counting packets unit has to be added as an auxiliary unit that counts the packets before they get streamed to the network.

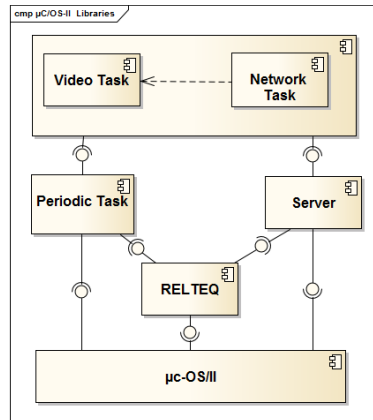


Figure 77-Enforcement Units Integration to Camera [18]

Figure 77 the Enforcement Units' integration on the camera is given. There are three main units in order to make the tasks (Video Task and Network Task) periodic: Periodic Task, Server and Relteq.

The system works as follows:

The period of a task is set (it did not have a period before). The period of the tasks can be changed later on. When the connection is published and the period of the task is sent by the AC. The period of the tasks are set within the RTSP_play (where the messages are received from the AC) by using the OSTaskSetPeriod() function. In addition, any task can set the period of any other task.

The role of the Relteq is to release the tasks, i.e. it marks them as a Ready task. Then the scheduler makes sure that at any moment the highest priority Ready task runs.

RE is based on the depletion event, which is inserted into server's virtual queue at every replenishment. When the event expires, the state of the server is changed to "depleted" and the server is switched off. The depletion event can be considered as part of monitoring which is the ability for tasks to check the remaining budget of a server. Then there is a method called `RelteqServerRemainingBudget()` which handles a server as a parameter and returns its remaining budget.

During the test, the results of the worst-case measurements of the task time period and the number of packets are manually recorded in the `LookUpTable`. The time period of the tasks are read from the `LookUpTable` and send along with the other unit messages (AC acceptance, delay and the number of packets).

However, after the RE was implemented; it did not function correctly on the camera. We assume that the reason is due to unstable camera software.

7.3.4 Delay Unit

The Delay unit needs an auxiliary component which is called the network packet sniffer to record the first JPEG packet that is received from the first camera and to record its arrival time should be recorded into the Registration List. While the second connection is requested, it is checked whether the overlap occurs in the first connection streaming period or not.

The sniffer is not implemented, because it is not used in the system and there are a lot of sniffer software applications to be found on the internet. Note that in order to give delay to the second camera, the streaming time of the first camera has to be known and two cameras have to be connected to the PC. For this reason, without using the sniffer, we assume that we have the connection time of the first camera and record it to the Registration List manually.

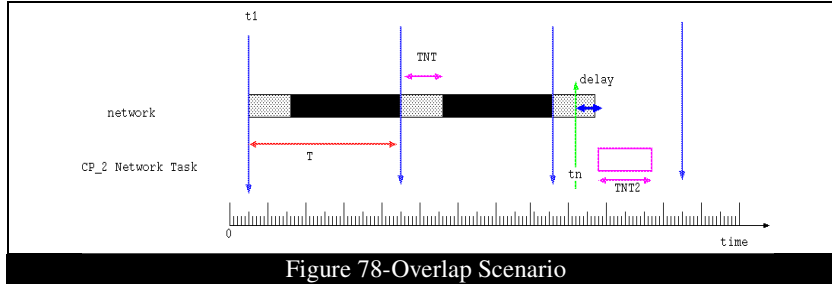
There are two changes that need to be made by the software developer: (1) semaphores (2) size of the buffer.

The first change is that, when the Video Task is divided into the two tasks (Video and Network tasks), the system starts to run the Video Task first, without an interruption of the Network Tasks (its semaphore flag is 0 at that moment). After this, the Network Task starts to run until it sets the Video Task semaphore flag to 1. In order to continue running the Video Task while the Network Task is sleeping, its semaphore flag needs to be set to 2 unlike the previous one which was 1.

The second change takes place in the original software; the size of the encoded frame buffer was set to 1 and is suitable for only one frame. Because of the added feature delay however, it is set to 2. The new size of the buffer is enough for the second frame while the first encoded frame is still within the buffer. Since increasing the second camera phase should not exceed the time interval between the two streamed frames from the first camera, new allocation should be enough for the second frame during that period of time, see Figure 59.

The synchronization between the buffers is provided by Message Queues. The input parameters of the Network Task: initial address of the frame and the size of the frame also take away the disorder between the streamed frames. Note that, frames are sent in a specific order and that the Network Task can only start when the input parameters are ready.

Figure 78 illustrates the terms, in order to explain the delay algorithm.



The terms are explained in Table 30.

Table 30-Delay At Once Abbreviations	
t_1	This is the time when the first registered camera JPEG packet is received. Note that there is only one camera that is why it is called “first camera” system.
T	This is the total streaming period of the first camera.
TNT	This is the time period of the Network Task for the first camera.
t_n	This is the expected time when the second camera starts
$TNT2$	This is the time period of the Network Task for the second camera system. Note that there are two cameras that is why it is called “second camera” system.
delay	This is the calculated time period the second camera has, see the equation below.

Equation 5 is given in order to show how the delay algorithm works. The “delay” variable is used in the decision of how long the second connection should wait before the frames start the streaming to the network.

$\text{If } (t_n - t_1) \% T < TNT$ $delay = TNT - [(t_n - t_1) \% T]$
Equation 5-Compute Frame Bit Rate

The phasing of the second connection depends on the first connected camera, and its initial streaming time.

The delay information is sent by the Socket Handler along with the AC request acceptance message. On the camera side `OSTaskSetPeriodEx()` function is used to set the initial offset. [18]

7.3.5 Scheduling

The scheduling is used to decide the resource assignments between possible tasks. In order to achieve scheduling, we first attempted to install the Real-time Operating System (RTOS) on the system nodes.

On the PC side RTOS is the RT-Preemption Patched in Linux which provides the priority preemptions feature to the tasks. The tasks can have the same priority level conversely in $\mu C/OS-II$. In this project, a number of VideoLAN (VLCs) are directly proportional with the number of the connected cameras. For this reason, each VLC has the same priority level, except for the non-real-time VLC. VLC can be scheduled either as Round-Robin (RR) or as First-In-First-Out (FIFO).

A real-time task priority is given to VLC and the FIFO scheduling method is chosen. However, it could not be tested, because the camera software only runs in debug mode, and the debug mode only runs in Windows (VMware).

7.4 Main Software Problems

In this section, the unstable software is explained along with the error messages. The main problem is concerned with *flashing the software* meaning that the compiled software is not uploaded. The original software developers initiated a

project, in order to erase the flashed firmware on the camera. However, even though the firmware was erased from the camera and the newly compiled software was uploaded, the camera did not process and received an IP “169.254.7.151”. This IP address indicates that the camera cannot get an automatic IP address, and that it is not possible to provide a static IP to the camera.

The following list gives the other problems we faced with the camera software:

1. *Immature software*: The semi-tested/unreleased code on the camera is immature. The streamed frames are received on the PC as shown in Figure 79.
 - o Reason:

When not all the packets were received, the JPEG algorithm filled the frame with a green block. The frame depicted in the figure is received from 1-camera system; meaning that there is no interference in the network. Nevertheless, the frame quality is low indicating that there is no packet loss on the network either. The only explanation could be that the packets are lost inside the camera.

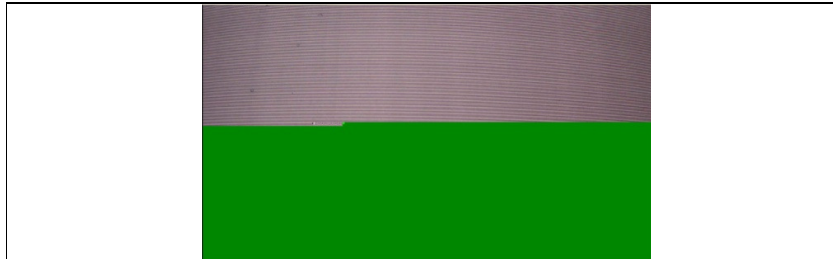


Figure 79-Incomplete Received Frame

2. *Error Message*: When we run the software on the camera up to a minute, we received an error as shown in Figure 80.

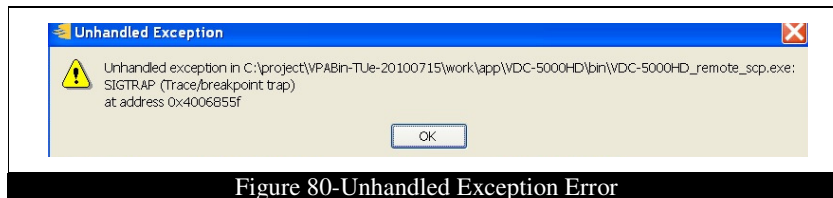


Figure 80-Unhandled Exception Error

3. *Development Environment*: The software development environment, Stretch IDE, requires a machine-dependent license. For this reason, we tested the entire implementation on Windows XP which was installed on Virtual Machine (VMware).
4. *Temperature*: The increased temperature of the camera is another observation we made. This increase could be caused by the fact that the top of the camera was removed in order to plug-in the JTAG print head. This can explain the packet loss as well because the software can very well be affected by the temperature.
5. *Color*: In high quality images, pink and blue strips are observed in the received frames, even though there are no pink or blue colors displayed on the captured frame.
6. *Calibration*: When the software is running in debug mode, the camera stops streaming when the calibration of the lens is changed.
7. *Web Interface*: Before we started to work with the camera software, we changed the IP address of the camera and the frame settings by using the camera web interface. However, after we started working with the new software, either the web interface just appeared for a couple of seconds, or the changes we made to the camera system, such as changes to the IP of the camera, were not reflected.
8. *Backup Software*: This software is used in order to recover the camera settings or changes to the IP address of the camera. In order to upload the software, the

catapult, camera and PC have to be in the same network subnet mask. However, the backup software did not work either. We were able to open the web interface by using the backup software, but we were not able to set a new IP, even when the IP seemed changed.

9. *Development Environment:* Compilation time of the camera firmware, uploading and the firmware takes too much time. The upload speed of the new ordered Catapult was slower than the broken one. However its speed varies on different computers. The speed of the catapult made the test cases difficult. The speed problem was not solved; even we contact and give all the details about the Catapult provider.(41)

8. Conclusion

This chapter presents the overall conclusions of the “A Real-Time Networked Camera System” project. Section 8.1 states all the conclusions, section 8.2 provides some recommendations and suggests a possible future extension of the current project.

8.1 Results and Conclusions

Three main goals of the current project have been defined:

- Have a distributed platform supported by a Real-Time Operating System (RTOS) on each node; the setup should consist of two cameras and a PC. The cameras should transmit streams to a PC.
- Have an example application that shows the resource management in a distributed context, two cameras and the PC.
- Have a protocol for communication between the real-time Kernels, this protocol should enable the integration of real-time communication and distributed control in order to admit system-wide decisions.

The first two goals have been achieved and the last goal has been partially achieved by providing Resource Management units: the Admission Control, the Bandwidth Availability, the Resource Enforcement and the Delay units.

The distributed system setup is constructed from a set of relatively independent components: two security cameras, a PC, and the network switch; the two cameras and PC are connected to each other via a network switch.

The video processing application is distributed to the system nodes. When the camera is turned on, it captures a raw frame, and encodes this frame. Whenever it is connected to the PC via the media player, it packetizes the encoded frame with the network protocol headers and streams it over the network to the PC. The PC receives the packets, removes the network protocol headers and reassembles the encoded frames, after which the frame is decoded and displayed on the screen.

The connection between the camera and PC is provided via the RTSP streaming protocol and the frames are streamed via the RTP protocol. The frames are encoded and decoded, based on the MJPEG algorithm.

In order to have a real-time distributed system, the PC is equipped with the RT-Preempt Patched Linux Kernel and is patched to the Ubuntu Linux distribution; the camera was already equipped with the μ C-OS/II real-time kernel. So, to complete the video processing chain which starts on the camera (capture-encode-stream), the PC is equipped with the VLC media player.

In order to manage the resources (processor/ bandwidth) and come to a system-wide decision, the Resource Management units (the central unit of the distributed system) are deployed to the PC. The communication between the Resource Management units and the camera is provided via a new proposed protocol called Management Protocol. This protocol is run on top of the connection-oriented (TCP) protocol, because in order to give a coherent behavior between the reserved resource and the current resource, the given decision should be connection oriented.

So, as to make a decision, there should be a decision criteria defined. For this reason, worst-case measurements are made. The purpose of the worst-case measurements is to create a real-time behavior with respect to the deadline of the tasks.

Due to the encountered software problems, the Resource Management unit is partially applied to the system. So, based on the Resource Management unit, the following results are provided:

- *The Admission Control unit:*
The bandwidth mainly utilized because of the packet streaming from the camera to the PC, and the number of the streamed packets depends on the quality of the frame. Quality refers to the perceptible visual quality of an image. The Admission Control unit adjusts the quality level of the frame based on the current bandwidth capacity.
- *The Bandwidth Availability unit:*
When the Admission Control unit allows the camera to stream a frame over the network to the PC, the system should check if the allowed connection still exist or not. The purpose is to keep the current utilized bandwidth up-to-date.
- *The Resource Enforcement unit:*
This unit is used to ensure that the tasks within the camera application do not utilize more resources than necessary. In contrast to the first two units, the Resource Management unit could not be tested on the camera because of the software problems. However, it was tested on the Stretch IDE simulator and it appeared to work. Based on the given period of time for the tasks they run as long as they are allowed to run. In order to provide this feature to the system, the RELTEQ and HSF methods are used.
- *The Delay unit:*
This unit is used in order to prevent the possible packet overlap on the network, whenever the second camera is connected to the PC. Based on a possible overlap, the second camera streaming time is delayed and the packets are streamed while the first camera does not stream over the network. Just as the previous unit, the Delay unit could not be tested as well. The reason is the same: because of the software problems the second camera could not be connected to the PC and therefore the delay unit test could not be carried out.

In summary, time synchronization is a crucial component in providing synchronization between the distributed system nodes. The possibilities for time synchronization are stated. However, none of them is neither designed nor implemented.

8.2 Recommendation and Future Work

The main goal of this project was to distribute the video processing application on the camera and PC, and schedule the video processing application collectively in real-time manner. However, due to an immature camera application and the lack of time, we only focused on the camera application.

If the software providers involved in this project could have done more than they actually did, this project could have reached a different level, in such that all the proposed units could have been applied, and even more design could have been proposed.

Based on the research we provided on the system the following recommendations are made for future work:

- Since the main camera system is defined in this project future developers do not need to spend as much time in understanding the system en in understanding on how the camera works.
- Since this project has provided the possible problems and proposed solutions, future developers can invest in solution improvements and unexecuted unit tests using the Resource Enforcement and Delay units.
- Since this project has enabled the task tracking system, it is possible to port to the µc-OS/II real-time kernel and the RT-Preempt patched Linux. The

project has provided software for the μ c-OS/II and documentation and references for the RT-Preempt patched Linux. The advantage of task tracking is to see whether the re-designed system behaves as expected. This advantage can be used in future projects.

- This project has enabled the optimization of the camera application with respect to the speed (memcpy to DMA); this can be an advantage for future projects in continuing to develop the current system based on the optimized camera system. Hence, the latency can be reduced.
- Since this project has determined time synchronization as a crucial component in providing synchronization between the distributed system nodes, future projects could focus more elaborately on the time-synchronization as such.

9. Project Management

This chapter introduces the various issues that are relevant to project management. The process used to manage the project is described in the first part. Other related subjects such as Breakdown structure, Milestone Trend Analysis, and Risk management are also presented in this section. A short retrospective of the project closes the chapter.

9.1 Process

This project can be characterized as a fixed-time and fixed-resource project. The project duration was 9 months, in which one project member had the full responsibility of the system design and development.

The project steering group consisted of two members, who had the primary responsibility of supervising, mentoring and advising the project member during the software design and development of the project.

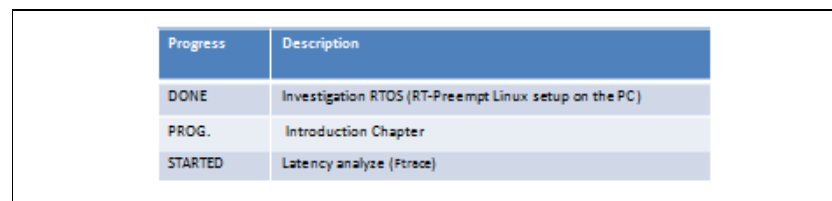
The project had a frequent communication pattern established from the beginning. Three different meetings were scheduled: one with the project steering group, one with the project group leader and one with the project owner.

- In the first two months (January and February), a weekly plenary meeting involving the project steering group took place in order to check the progress of the project and the processes involved.
- During the project, a weekly meeting involving the project group leader from the SAN group was scheduled in order to brainstorm, take decisions, show the progress of the project and receive feedback and come to an agreement.
- During the project, a weekly planned but monthly executed meeting involving the project owner from TU/e was scheduled in order to brainstorm, take decisions, show the progress of the project and receive feedback and come to an agreement.

9.2 Planning and Tracking

During the project some tools were used: the Microsoft PowerPoint tool for the weekly presentations, the Microsoft Office Excel tool to create the milestone trend analysis and the Microsoft Office Project tool for arranging the project planning.

During the project, the Microsoft PowerPoint tool (Figure 81) was used in order to show the status of the tasks: DONE, IN PROGRESS, STARTED.



Progress	Description
DONE	Investigation RTOS (RT-Preempt Linux setup on the PC)
PROG.	Introduction Chapter
STARTED	Latency analyze (Ftrace)

Figure 81-Weekly Meeting Presentations

The Microsoft Office Project tool (Figure 82) was used as a project management tool. The project is divided into two main parts: global and short iteration. The short iteration plan contains a description of the tasks needed to tackle the requirements assigned to the iteration. The global design is divided into four main parts: after the first part (background information, project management and feasibility analysis), the following three parts have been divided to contain the same structure as in design, implementation, test and documentation. However, the planning parts were not as easy as making the plan. A lot of drawbacks obstructed the plan to develop and test the proposed design.

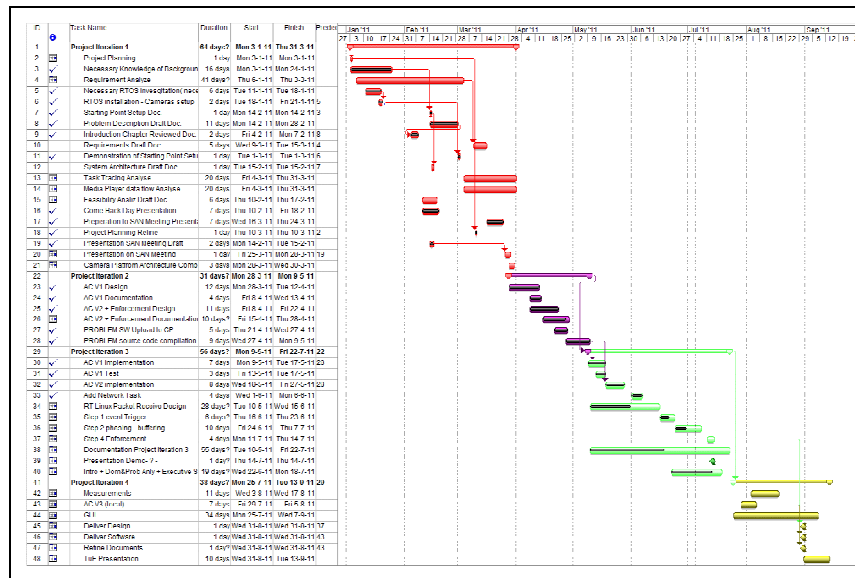


Figure 82-Microsoft Office Project tool, iterative planning

The Microsoft Office Excel tool (Figure 83) was used as a Milestone Trend Analysis (MTA). First three months of the project MTA was used and short term plans were followed:

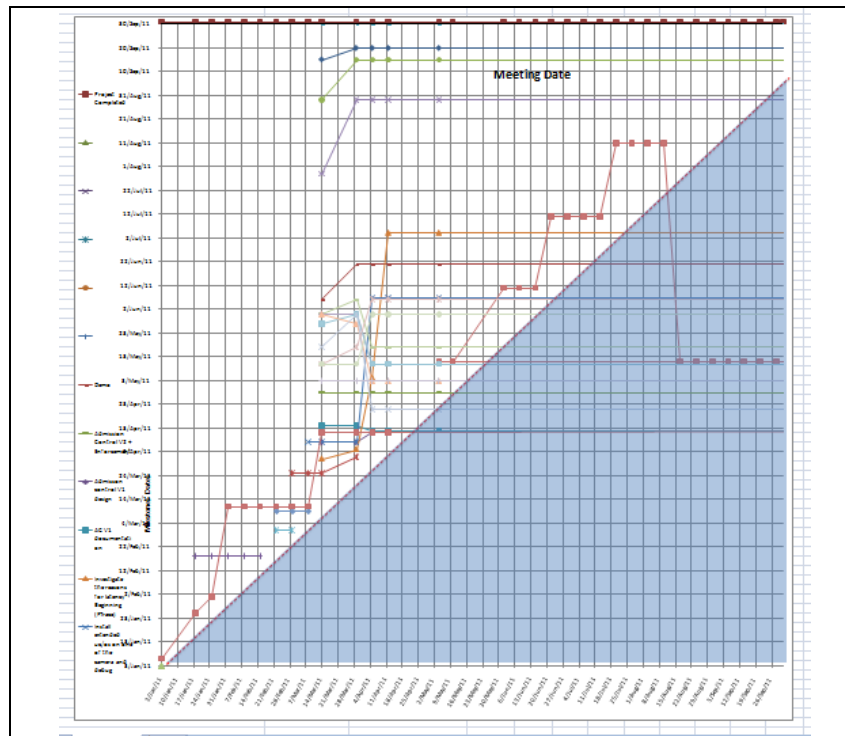


Figure 83-Microsoft Office Excel Milestone Trend Analysis

9.3 Risk Management

The benefits of risk management in projects are huge and like all software development projects, also this project involved risks. These risks were identified at the beginning of this project and during the project. In Table 31 the most important risks are described.

Table 31-Most Important Identified Project Risks

<i>Risk</i>	<i>Description and Mitigation</i>
Learning Curve	The lack of experience in the embedded system and the specific camera platform were quite a challenge. Regardless of this, there is always the risk of going into the wrong direction. That is why it is essential to regularly summarize the findings to the supervisors.
Project Planning	Unexpected situations altered the planning. The project planning needed to be adjusted based on the new situation.
Communication Problem	Language and background differences may have caused misunderstandings between the author and stakeholders. This was avoided by having the author provide a progress report during the meetings on the actions taken.
Summer Holiday	People involved in the project are unavailable during holidays. In order to anticipate, the project planning included the agenda of the supervisors.
Imported Device	Next to the camera, also the catapult, which was ordered from the USA, was used in this project. The catapult was broken and before ordering a new one, an attempt was made to fix it with the help of the company it was ordered from. However, because of the different time zone, working hours and sleeping hours are not the same. The only way to solve this was when one of the partners sacrificed some sleeping time. Eventually we were not able to fix it and ordered a new catapult, so a lot of time was lost during the attempt to fix it and receive a new one.

9.4 Retrospective

In retrospect, a lot of lessons can be learned from this project. Therefore, an overview of the positive experiences and these lessons are presented in the following list so that future projects may benefit:

- Finding experts in the organizational environment is very important. They can help to find potential problems earlier and not during the technical implementation. In this project they proved to be extremely supportive and helpful.
- Working with the third parties has to be considered as a big risk. They have their own businesses and may not have enough time to provide support to external projects. This was experienced in this project.
- Using imported tools is a big risk when they are broken, because trying to fix the tool before ordering a new one and the transportation time after ordering waists a lot of time as experienced in this project.
- Even if the prototype is simple, it is better to demonstrate it to the stakeholders, in order to avoid unexpected situations.
- Working in a structured way can highly improve the productivity.
- When the project is in an early stage and many things are not yet clear, it is very important to involve as many experts as possible to have technical discussions.
- Writing the final report should be a continuous process during the entire project. It should be taken into account that during the summer, most of the supervisors are on holiday so the reviewing time is very limited.
- Be conscious of possible misunderstandings between parties; keep asking questions and report on a regular basis on the progress of the project.
- Be aware of the development environment. For the embedded systems, software uploading and compiling processes can take more time than estimated. For this reason, without being sure, these processes should not be started.

While designing a project, there are ten technological design criteria that can be taken into account: functionality, impact, possibility of realization, inventiveness, complexity, elegance, genericity, methodical approach, and documentation. Not all these criteria are applicable for all projects since their level of importance varies. In this project, three criteria have been chosen as the most important for this project. A brief description of how they were achieved is presented below:

- **Functionality:**

This project started as an idea and a fixed goal. It was very important to determine how that idea could adhere to the system and what could be achieved. A set of functionalities that would be desirable were defined before we started the project. However, the functionalities for the resource management and scheduling were only based on terminology not practiced. The research focused on how the functionalities could be achieved and could be designed trying to accommodate all of them. At the end of the project possibilities were provided and the requirements were achieved as much as possible.

- **Possibility of Realization:**

The project was design-oriented and started with some vague requirements. The feasibility study was the most important phase of the project and it tried to discover if and how it is possible to define the functionality. The design focused on the possibility to combine the findings from the feasibility study in order to provide the required functionality. This project is based on State-of-the-Art technologies. All the technologies and building blocks already exist. Therefore, it was possible to implement the design commercially at low cost.

- **Genericity:**

The re-designed system combines as many as possible standard technologies and tools that prove to be reliable and efficient. The purpose of the implemented solution is to provide something simple, easy to maintain and effective without “reinventing the wheel”. It also provides components that can easily be reused in further developments. During the feasibility study and the design elaboration, a set of practiced solutions and several already existing similar solutions were identified. Resource management units can be used not only for the bandwidth but also for the other resources: memory and processor.

The two out of the ten design criteria that are less important are:

- **Inventiveness:**

The project tried not to reinvent the wheel and many standard tools and technologies are used. The design is inspired by different success-proof content delivery designs, thus the inventiveness of the implemented solution and its design is rather low.

- **Elegance:**

The project aimed to be a proof-of-concept. For this reason, the elegance of the design was not considered as an important criterion. However, the re-design of the implemented part of the solution is extensible with well-defined layers and components.

Bibliography

- [1]point.onehttp://www.point-one.nl/Service_menu/English
- [2]Omeca: Optimization of Modular Embedded Onderdeel van Subsidieaanvraag Point One 2009
- [3]System Architecture and Networking<http://www.win.tue.nl/san/>
- [4]. **Z. Deng, J. W.-S. Liu, J. Sun.** A Scheme for Scheduling Hard Real-Time Applications in Open System Environment.
- [5]*Concepts and diagram elements for architectural* 2011
- [6]. **Rajarajan, D.** Porting of Micro C/OS-II kernel in ARM powered microcontroller.
- [7]. **Alan Burns, Andy Wellings, Stephen A. Edward, Kang G. Shin.** RTOS – Real-Time Operating System.
- [8]. **Stankovic, John A.** A Series Problem for Next-Generation Systems.
- [9]*Embedded Operating Systems for Real-Time Applications* IT Bombay November 2002
- [10]. **Baker, T. P.** Introduction to Periodic Tasks. <http://www.cs.fsu.edu/~baker/realtime/restricted/notes/periodics.html>. [Online]
- [11]. **Thiele, Lothar.** Embedded Systems, Aperiodic and Periodic Tasks.
- [12]Micrium<http://micrium.com/page/home>
- [13]*Handbook of Image and Video Processing* 1033
- [14]. JPEG. <http://www.jpeg.org/jpeg/index.html>. [Online]
- [15]*Architecture of Distributed Systems/ Lecture Notes* 2011-2012
- [16]. **Sojka, Michal.** Resource Reservation and Analysis in Heterogeneous and Distributed Real-Time Systems. 2010.
- [17]. http://en.wikipedia.org/wiki/Temporal_isolation. Wikipedia. [Online]
- [18]*Extending an Open-source Real-time Operating System with Hierarchical Scheduling* Technical Report, Eindhoven University of Technology October 2010 CS-10-10
- [19]. **Rashid, Alex Gantman Pei-Ning Guo James Lewis Fakhruddin.** Scheduling Real-Time Tasks in Distributed Systems: A Survey .
- [20]. http://www.artist-embedded.org/docs/Events/2009/OSP/OSP09_Holenderski.pdf. [Online]
- [21]. **Kopetz, Hermann.** Real-Time Systems: Design Principles for Distributed Embedded Applications .
- [22]*Hierarchical Scheduling of Complex Embedded Real-Time Systems*
- [23]*Extending a HSF-enabled Open-Source Real-Time Operating System with Resource Sharing*
- [24]Internet Technical Resources<http://www.cs.columbia.edu/~hgs/internet/>
- [25]About the Real Time Protocolhttp://toncar.cz/Tutorials/VoIP/VoIP_Protocols_Real_Time_Protocol.html

- [26]Javvin Network Management and Security<http://www.javvin.com/protocolRTCP.html>
- [27]*Identifying bottlenecks in the performance of an existing surveillance application*August 4, 2011
- [28]. **Elke Hochmüller, Michael Dobrovnik.** *Identifying Types of Extra-Functional Requirements in the Context of Business Process Support Systems.*
- [29]*Real-Time and Embedded Guide* K.U.Leuven, Mechanical Engineering2000
- [30]OSADL Project: "Latest stable" RT-Preempt realtime Linux kernel<https://www.osadl.org/Latest-Stable-Realtime.latest-stable-realtime-linux.0.html>
- [31]*Porting RT-preempt to Loongson2F* Tianshui South Road 222,Lanzhou,P.R.China
- [32]. **Jones, M. Tim.** Anatomy of real-time Linux architectures. <http://www.ibm.com/developerworks/linux/library/l-real-time-linux/>. [Online]
- [33]. RT PREEMPT HOWTO. <http://www.mail-archive.com/linuxkernelnewbies@googlegroups.com/msg01769.html>. [Online]
- [34]. Tech Radar - The best media player for performance 2011. http://www.techradar.com/news/software/applications/the-best-performing-media-player-for-2010-683569?artc_pg=2. [Online]
- [35]The architecture of VLC media framework<http://www.enjoythearchitecture.com/vlc-architecture>
- [36]. linuxkernelnewbies . <http://www.mail-archive.com/linuxkernelnewbies@googlegroups.com/msg01769.html>. [Online]
- [37]Wireshark Official Website<http://www.wireshark.org/>
- [38]. http://en.wikipedia.org/wiki/JPEG#JPEG_codec_example. [Online]
- [39]. Cygwin. <http://www.cygwin.com/>. [Online]
- [40]. **Parameswaren Ramanathan, Kang G. Shin, Ricky W. Butler.** *Fault-Tolerant Clock Synchronization in Distributed Systems.*
- [41]ByteTool<http://www.byte-tools.com/>
- [42]. **Verhagen, Norbert.** *Setting up the hard and software.* 2010.
- [43]. <http://www.ndmp.org/wp/wp.shtml>. [Online]
- [44]. http://www.ndmp.org/info/spec_summary.shtml. [Online]
- [45]Image Resolution, Size and Quality<http://www.microscope-microscope.org/imaging/image-resolution.htm>
- [46]Rowe Bots RTOS Products for DSCs DPS and Microcontrollershttp://www.rowebots.com/embedded_system_software/rtos
- [47]*TIES426 Real-time systems* FinlandFI-40014 University of Jyväskylä2009
- [48]*Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited*TU/e
- [49]UML 2.0 Notation for Modeling Real Time Task Scheduling<http://www.uml.org.cn/UMLApplication/200612264.htm>

- [50] Multi-processor Motion JPEG decoder http://www.es.ele.tue.nl/mininoc/doc/mjpeg_mmips.htm
- [51] *Real Time Communication in Embedded Systems* 2010
- [52] **Benjamin Wah.** *Wiley Encyclopedia of Computer Science and Engineering*
- [53] *Real-Time Operating System* Kluwer Academic Publisher 2004
- [54] *Holistic Scheduling and Analysis of Mixed Time/Event Triggered Distributed Embedded Systems* Linköping University, Sweden
- [55] *Resource Kernels: A Resource-Centric Approach to Real Time and Multimedia Systems*
- [56] *Swift mode changes in memory constrained real-time systems*
- [57] *Towards Hierarchical Scheduling in VxWorks*
- [58]. **Hideyuki Tokuda, Tatsuo Nakajima, Prithvi Rao.** *Real-Time Mach: Towards a Predictable Real-Time System.*
- [59]. http://en.wikipedia.org/wiki/Distributed_computing. [Online]
- [60]. NixCraft. <http://www.cyberciti.biz/faq/howto-set-real-time-scheduling-priority-process/>. [Online]
- [61]. **Chaudhury, Dr.Santanu.** Lecture -1 Embedded Systems: Introduction.
http://www.youtube.com/watch?v=y9RAhEfLfJs&list=FLAdpcuj0HJ1_cYm dSZtPoSw&index=1. [Online]
- [62]. Cisco Unified Fabric.
<http://searchnetworking.techtarget.com/definition/round-trip-time>. [Online]
- [63]. https://svn.win.tue.nl/repos/ucos/apps/example_period_task. [Online]

Appendix-A

Table 32 shows the criteria for the selection of the RTOS selection with the description, advantages, and disadvantages of each criteria.

Table 32-Real-time Operating System Criteria

Term	Definition	Advantage	Disadvantage
Latency	Measure of time delay experienced in a system, the precise definition of which depends on the system and the time being measured.		
Context Switch	Changing the task processing on the CPU.		
Dual Mode	One part of the system for high data load (includes DSP), another part supports priority based handling.	Developer can optimize the system without having to purchase a specific high data load system.	
App. Power consumption restriction		Interrupt Driven RTOS saves the power, and extend battery life.	
Tick for per time management			Processor to be frequently activated which consumes additional power.
Disable interrupt		Protect the internal structures.	Take hundreds of cycles and limits the response time.
Interrupt Stack and Fiber support	New generation architecture supports for optimal RAM usage.	Save precious RAM	
Multithread Thread			
Global pointers			Lead the corruption and requires management.
External-Internal Clocks	Internal keeps track of RTOS operations.		Their synchronization is a crucial issue.
SW timer		One-shot and periodic sw timers can be used by multiple threads	
Protection Mechanism	Mutual exclusion, counting, semaphores, critical sections.		
Source code		User can minimize	

Term	Definition	Advantage	Disadvantage
availability		the risk.	
Documenta- tion		Easy to use	
Hardware Specific		Interrupt architec- ture and compilers to its fullness.	Usage of RTOS is limited.
Error han- dling		Easy to debug.	
Processor Support			If not existing, porting the RTOS to that processor is necessary.
Scheduling			Overhead
Preemption			Overhead
Licensing			

Table 33 shows the Selected RTOSs and their features. RT-Preempt Patched Linux suits this project the most.

Table 33-Selected RTOS							
Possibilities							
Sync.	Arch.	Type	License	Source	Documentation	Version	Name
Mutex	x86	Patch2.6.33.7.2-rt30.bz2	Free	Open	Web-site	Latest stable:2.6.33	RTPreempt
	x86	Debian Kernel	Free	Precompiled	Web-site INADEQUATE	Last modified 28-dec-2010	Pengutromix
		Patch	Free	Open	Web-site	2.5.5.2	Xenomai
Mutex, semaphores	i386, MIPS, PPC, ARM	Kernel	Not clear	Open	Web-site	3.8	RTAI
Mutex, semaphores	i386, PPC, ARM	Kernel	Free /not-Free	Open	Web-site INADEQUATE		RTLinux
	Intel x86-32 and x86-64	Patch (soft real-time)	GNU-GPL	Open	Web-site NOT STABLE	2010.02	LITMUS _{RT}
		Patch (network) RTAI-Xenomai compatible		Open	Web-site		RT-NET

Debugging

The debug mode is basically showing the state of the software on the camera platform. The reason to discuss the debug mode is that the worst case measurements are gathered with the camera application running in the debug mode, because the camera application software was so unstable that it did not run in the release mode.

The software on the camera (embedded system) is compiled on the PC and uploaded to the camera before execution. For these processes hardware (catapult, JTAG, and serial cable) and software (Stretch IDE, Xtensa OCD Deamon, and HyperTerminal) is needed. The next two sections give information how the camera is prepared for the debug mode. For more detailed information about the setting of the software and hardware see [42].

Debugging Hardware Tool

The camera and the PC are connected to each other via a tool: Catapult EJ-2 Ethernet-to-JTAG device (41). The Catapult is used to upload software from PC to the camera via the JTAG interface. The PC is connected to the Catapult via an Ethernet cable and the Catapult in turn is connected to the camera via the JTAG interface. This JTAG interface differs from embedded system to embedded system. Figure 84 shows the Catapult and the JTAG connection.

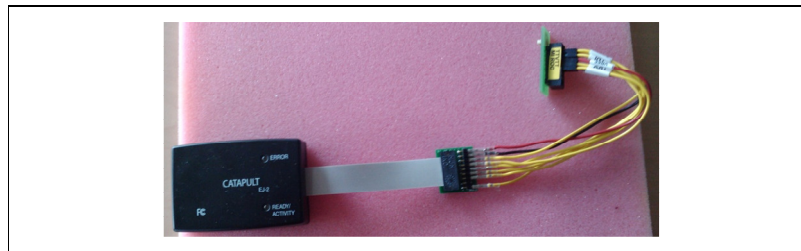


Figure 84-CatapultEJ2-Ethernet –to-JTAG device: Yellow Wires are called Flying Leads, and Black-Green Head is called JTAG Print Head; JTAG Print Head is plugged to Camera

The software is uploaded to the camera after it is compiled on the PC and the state of the camera is monitored on the PC side. During the debugging process the software is temporarily located on the camera, i.e., if the debugging hardware is turned off or the debugging software is closed, then the uploaded software will be removed from the camera. Nevertheless, it is possible to upload the software to the camera permanently.

There is one more debugging tool which is the Serial Cable connection which provides a direct connection between the camera and the PC. For debugging purposes the developer annotates the source code with printf statements to get additional information on the state of the camera. Using the Serial Cable connection this information is displayed by means of a HyperTerminal which is installed on the PC.

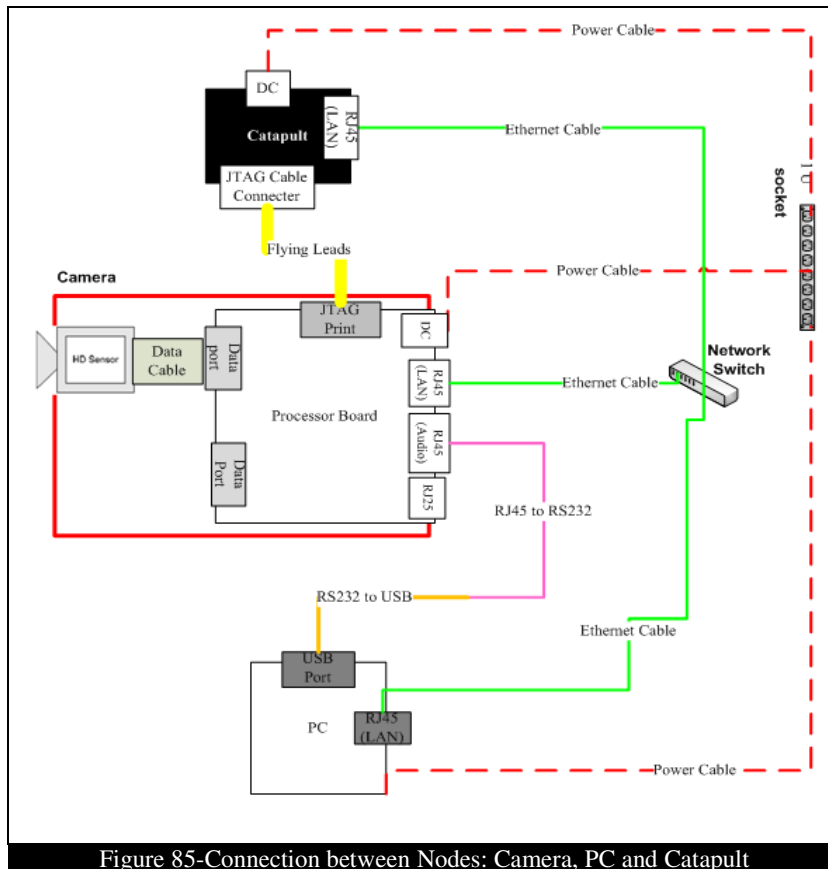


Figure 85-Connection between Nodes: Camera, PC and Catapult

Figure 85 shows the hardware needed for the debugging: CatapultEJ2-Ethernet – to-JTAG, Ethernet Cable and Serial Cable which has two parts because there is no direct connection from the RJ45 to the USB port. For this reason first the Ethernet to the RS232 cable is used and it is connected to the RS232 to the USB cable and the USB jag is connected to the USB port on the PC.

Debugging Software Tool

Software development for the application on the camera, the Stretch Integrated Development Environment (Stretch IDE) is provided by the video application developers of the camera.

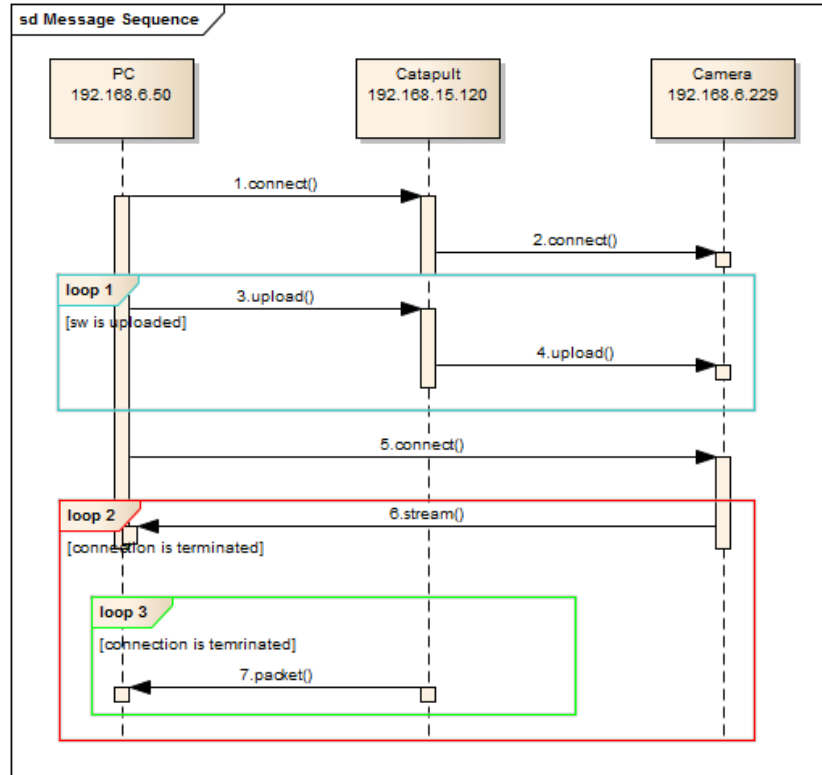


Figure 86-Message Sequence among PC-Catapult-Camera: Camera Application Upload and Streaming

Figure 86 shows the message sequence among the nodes during the software upload and the streaming. After the nodes are connected physically, see Figure 85, the PC sends a connection request to the Catapult and with Stretch IDE the software is uploaded from the PC to the camera over the Catapult via the JTAG cable.

Loop 1 presents the software upload. After the software is uploaded a connection request is sent directly from the PC to the camera, where after the camera application starts to stream the frames to the PC.

During the period of Loop 2, it is observed that the Catapult sends NDMP (network data management protocol) messages to the PC: “NDMP is a network protocol that specifies the communication between the server and the backup software. Communication is defined using a series of defined interfaces. (43)The purpose of this protocol is to allow a backup of a host without requiring a full port of the backup software product. The backup and restore solution is partitioned in a way that minimizes the amount of software required on the host.”(44)

About the Author



Hilal Karatoy studied Information Technologies as a major study and Computer Science and Engineering as a minor study at the Isik University in Istanbul (Turkey). She got a full scholarship for Bachelor's Degree program (5 years) and Master's Degree program (2 years) studies. She received her Master's Degree in Computer Science and Engineering at the Isik University in Istanbul (Turkey) in December 2009. During the master studies she worked as a Teaching Assistant and 2009 she decided to join Stan Ackermans Institute (SAI). Her research interests include computer networks, computer security, real-time embedded systems, and bioinformatics.

3TU.School for Technological Design,
Stan Ackermans Institute offers two-year
postgraduate technological designer
programmes. This institute is a joint initiative
of the three technological universities of the
Netherlands: Delft University of Technology,
Eindhoven University of Technology and
University of Twente. For more information
please visit: www.3tu.nl/sai.