

UNIVERSITY OF CALIFORNIA,  
IRVINE

**Cooperative Cross-Layer Protection for Resource  
Constrained Mobile Multimedia Systems**

DISSERTATION

submitted in partial satisfaction of the requirements  
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Kyoungwoo Lee

Dissertation Committee:  
Professor Nikil Dutt, Chair  
Professor Nalini Venkatasubramanian  
Professor Lichun Bao

2008



The dissertation of Kyoungwoo Lee  
is approved and is acceptable in quality  
and form for publication on microfilm:

---

---

---

Committee Chair

University of California, Irvine  
2008

## DEDICATION

*To my family*

# Contents

<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>ix</b>
<b>Acknowledgments</b>	<b>x</b>
<b>Curriculum Vitae</b>	<b>xii</b>
<b>Abstract of the Dissertation</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Embedded Systems . . . . .	1
1.1.1 Pervasive Embedded Systems . . . . .	1
1.1.2 Mobile Multimedia Embedded Systems and Multi-dimensional Constraints . . . . .	3
1.2 Motivation and Objectives . . . . .	4
1.2.1 Errors and Failures . . . . .	4
1.2.2 Conventional Redundancy Techniques . . . . .	6
1.2.3 Resource Efficient Reliability . . . . .	8
1.3 Thesis Overview . . . . .	8
1.3.1 EAVE (Error Aware Video Encoding) . . . . .	10
1.3.2 CC-PROTECT (Cooperative Cross-layer Protection) . . . . .	11
1.4 Thesis Contributions . . . . .	11
1.4.1 Effectiveness of Cross-layer Approaches for Reliability . . . . .	11
1.4.2 Cost-efficient Reliability . . . . .	12
1.4.3 Expanded Design Space Exploration . . . . .	12
1.4.4 Extended Applicability of Existing Techniques . . . . .	13
<b>2 Cross-Layer Approach</b>	<b>14</b>
2.1 Case Study: Network Layering . . . . .	15
2.2 Cross-Layer Approach in Mobile Embedded Systems . . . . .	18
2.2.1 QoS/Performance/Energy Tradeoffs . . . . .	19
2.2.2 QoS/Power/Timing Tradeoffs . . . . .	21
2.3 Reliability across System Layers . . . . .	22
2.3.1 Reliability at the Hardware Layer . . . . .	23

2.3.2	Reliability at the Application and Middleware Layers . . . . .	24
2.3.3	Reliability at the Network Layer . . . . .	24
<b>3</b>	<b>Partially Protected Caches: Enabling Reliability at the Hardware Layer</b>	<b>26</b>
3.1	Motivation . . . . .	26
3.2	Related Work . . . . .	29
3.2.1	Soft Errors . . . . .	29
3.2.2	Soft Error Rate and Vulnerability . . . . .	29
3.2.3	Soft Error Protection Techniques . . . . .	30
3.2.4	Software Solutions . . . . .	33
3.3	Architecture of Partially Protected Caches (PPC) . . . . .	34
3.4	Page Partitioning Techniques . . . . .	35
3.4.1	Application Data Partitioning in Multimedia Applications . . . . .	35
3.4.2	Page Partitioning in General . . . . .	37
3.5	Effectiveness of PPC . . . . .	40
3.5.1	Experimental Framework . . . . .	40
3.5.2	Experimental Results . . . . .	41
3.6	Summary . . . . .	46
<b>4</b>	<b>Error Aware Video Encoding: Enabling Reliability at the Application Layer</b>	<b>48</b>
4.1	Motivation . . . . .	48
4.2	Background . . . . .	51
4.2.1	Energy/QoS-aware Video Encoding . . . . .	51
4.2.2	Error-Resilient Video Encoding . . . . .	53
4.2.3	Error-Aware Video Encoding: Our Proposal . . . . .	54
4.3	Error Aware Video Encoding . . . . .	55
4.3.1	System Model . . . . .	55
4.3.2	Fundamentals of Active Error Exploitation . . . . .	56
4.3.3	EA-PBPAIR: An Error-Aware Video Encoder . . . . .	57
4.3.4	Adaptive EAVE . . . . .	59
4.4	Effectiveness of EAVE . . . . .	60
4.4.1	Experimental Setup . . . . .	60
4.4.2	Experimental Results . . . . .	62
4.5	Summary . . . . .	69
<b>5</b>	<b>Cooperative Cross-layer Protection</b>	<b>71</b>
5.1	Motivation . . . . .	71
5.2	A Cross-Layer Approach to Support Reliability and QoS . . . . .	73
5.2.1	System Model and Problem Definition . . . . .	73
5.2.2	Related Work . . . . .	75
5.2.3	Cooperative Cross-Layer Approach . . . . .	76
5.3	Cooperative Cross-Layer Protection (CC-PROTECT) . . . . .	79
5.3.1	Error Detection at Hardware . . . . .	79
5.3.2	Drop and Forward Recovery at Middleware . . . . .	80
5.3.3	Error-Resilient Video Encoding at Application . . . . .	81

5.4	Intelligent Selective Algorithms . . . . .	82
5.5	Effectiveness of CC-PROTECT . . . . .	84
5.5.1	Experimental Setup . . . . .	84
5.5.2	Experimental Results . . . . .	89
5.6	Summary . . . . .	95
<b>6</b>	<b>Conclusion</b>	<b>97</b>
6.1	Summary . . . . .	97
6.2	Contribution . . . . .	98
6.3	Future Directions . . . . .	99
	<b>Bibliography</b>	<b>101</b>

# List of Figures

1.1	Pervasive Mobile Embedded Systems . . . . .	2
1.2	Errors and Redundancy Techniques at Each System Abstraction layer . . . . .	5
1.3	No cooperation has been studied to actively exploit existing error-resilient techniques across system abstraction layers. . . . .	7
1.4	Overview of Thesis Proposals – PPC, EAVE, and CC-PROTECT are resource efficient reliability techniques in a cooperative, cross-layer manner. . . . .	9
2.1	ISO OSI 7 Layer Model and Cross-Layer Proposals [109, 112] . . . . .	16
2.2	System Abstraction Layers for Mobile Embedded Systems and Related Work . . . . .	18
2.3	Examples of Errors and Redundancy Techniques at Each Abstraction Layer . . . . .	23
3.1	Overview of our proposal, PPC and unequal protection in a cross-layered manner . . . . .	28
3.2	External radiation may induce soft error . . . . .	29
3.3	PPC (Partially Protected Caches) - one protected and the other unprotected at the same level of memory hierarchy . . . . .	35
3.4	Failure rate distribution (benchmark : <i>susan edges</i> ) - failures are reported in occurrences of soft errors at only 9 pages out of 83 . . . . .	36
3.5	Size of failure critical and failure non-critical data in applications . . . . .	36
3.6	Cache miss rates of failure critical and failure non-critical data (benchmark - <i>susan smoothing</i> ) . . . . .	37
3.7	Vulnerability and Failure Rate: vulnerability is a good metric for estimating failure rate . . . . .	38
3.8	DPExplore: an exploration algorithm for data partitioning . . . . .	39
3.9	Experimental Framework . . . . .	40
3.10	Effectiveness of our PPC architectures - PPC achieves minimal failure rates with minimal energy and performance overheads . . . . .	41
3.11	Evaluation of Quality and Area . . . . .	43
3.12	Evaluation under <i>No Performance Penalty</i> and <i>5% Performance Penalty</i> : DPExplore can significantly reduce the vulnerability at minimal runtime and power overheads . . . . .	45
3.13	Cross-Layer Protection – PPC and Page Partitioning Algorithms provide cost-efficient unequal protection for resource-constrained embedded systems . . . . .	47



4.1	Overview of our proposal – EAVE (Error-Aware Video Encoding) . . . . .	50
4.2	Constraints and knobs considered by previous approaches and our proposal . . .	51
4.3	System Model (Mobile Video Conferencing) and Frame Drop Types I/II/III for Active Error Exploitation . . . . .	55
4.4	Error-Aware Video Encoder is composed of Error-Injection Unit and Error-Canceling Unit with a Knob (Error Injection Rate) . . . . .	57
4.5	Flow of Error Controller and Adaptive EIR in EA-PBPAIR for Mobile Video Applications . . . . .	58
4.6	Experimental Framework for Mobile Video Conferencing System - <i>System Prototype</i> + <i>NS2</i> Simulator . . . . .	60
4.7	Effects of Error Injection Rate on Energy Consumption and Video Quality in EA-PBPAIR compared to GOP-3 (PLR = 10%, FOREMAN 300 frames, Each encoding is constrained with bandwidth) . . . . .	63
4.8	Energy Reduction and Quality Degradation of EA-PGOP compared to PGO (PLR = 10%, EIR = 10%, Error rate is adjusted, FOREMAN 300 frames) . . . . .	64
4.9	Energy Reduction and Quality Degradation of EA-GOP compared to GOP (PLR = 0%, EIR = 20%, Error rate is adjusted, FOREMAN 300 frames) . . . . .	65
4.10	Extended Tradeoff Space between Video Quality and Energy Consumption by EA-PBPAIR in comparison to GOP-8 and PBPAIR (EIR = 0% to 50%, PLR = 5%, FOREMAN 300 frames, Each encoding is constrained with bandwidth) . . .	66
4.11	Adaptive EA-PBPAIR Robust to Varying PLR under Dynamic Network Status . .	68
4.12	Cross-Layer Error-Exploitation – EAVE maximally exploits the energy efficiency and error resilience of previously proposed video encodings by intentionally injecting errors for resource-constrained embedded systems . . . . .	69
5.1	Overview of our proposal, CC-PROTECT . . . . .	72
5.2	System Model - Mobile Video Encoding System . . . . .	77
5.3	CC-PROTECT (Cooperative Cross-layer Protection) – mitigating hardware defects with minimal costs by using error-resilience and a Drop and Forward Recovery (DFR) in a video encoding . . . . .	78
5.4	Error Recovery Mechanisms . . . . .	81
5.5	Intelligent Selective Schemes . . . . .	82
5.6	Experimental Setup – Compiler/Simulator/Analyzer . . . . .	87
5.7	CC-PROTECT achieves the low-cost reliability at the minimal QoS degradation .	90
5.8	Intelligent selective schemes maintain the video quality and reliability with minimal overheads of power and performance . . . . .	94
5.9	Cooperative Cross-Layer Protection – CC-PROTECT exploits existing error control schemes across system abstraction layers to achieve cost efficient reliability .	96

# List of Tables

3.1	Video Quality in PSNR ( <i>dB</i> ) according to SER . . . . .	43
4.1	Energy Reduction at the Cost of Video Quality . . . . .	67
5.1	Error models and error control schemes at different abstraction layers . . . . .	74
5.2	System Compositions - Our CC-PROTECT is a middleware-driven, cooperative approach aware of hardware failures . . . . .	85
5.3	CC-PROTECT is very effective in terms of performance, power, and reliability at the minimal QoS degradation for different video streams (normalized result of each composition to that of BASE) . . . . .	92

# Acknowledgments

I would like to take this opportunity to thank all the people who have contributed to this dissertation.

Foremost, I would like to thank my advisors, Professor Nikil Dutt and Professor Nalini Venkatasubramanian, for their excellent guidance and support during my PhD studies. They taught me how to think about problems, how to approach the solution, and how to present our work in literature and talk. Professor Dutt gave me ample freedom and infinite encouragement to explore my ideas, and Professor Venkatasubramanian gave me a lot of inspiration for our works. I will forever be grateful and indebted for the time and the effort they have spent in my education.

I would also like to thank Professor Lichun Bao for his guidance and for serving on my thesis committee. I would like to thank Professor Aviral Shrivastava for not only being my friend but also being my mentor. Discussion and collaboration with him gave me the confidence in my work and stimulated me to challenge new problems.

I would also like to thank many friends in the ACES lab and the DSM lab. In particular, I would like to thank Dr. Minyoung Kim, Dr. Ilya Issenin, Dr. Radu Cornea, and Dr. Shivajit Mohapatra for their collaboration. I would like to thank all the members in the CECS, who have provided stimulation, support, friendship, and more. Special mention must be made of the ICS staff, particularly Melanie Sanders, for all their help throughout the years.

This thesis would not have been possible without the support and patience of my wonderful wife, Dr. Hyunkyung Lee. She has been encouraging me even when I had a hard time, and presenting her respect, belief, and love on every step I have been reaching.

I would like to thank my parents and my family, to whom this thesis is dedicated. This thesis is a fruit of infinite love and sacrifices from my father Jongoh Lee and mother Hyunie

Chung. My brother Wonwoo Lee and sister Sunhwa Lee have always given me better chance and environments than they had. I would also like to thank the belief and support from Hyunkyung's family.

Finally, I would like to thank all my personal friends. Without them, life would be very tough and boring.

# Curriculum Vitae

**Kyungwoo Lee**

## **Education**

PhD in Computer Science, University of California at Irvine, Irvine, CA, USA, 2003 – 2008

MS in Computer Science, Yonsei University, Seoul, Korea, 1995 – 1997

BS in Computer Science, Yonsei University, Seoul, Korea, 1991 – 1995

## **Summary of Work and Research Experience**

Graduate Research Assistant, School of Information and Computer Science, University of California at Irvine, CA, USA, 2003 – 2008

Teaching Assistant, School of Information and Computer Science, University of California at Irvine, CA, USA, 2003 – 2006

Research Engineer, Digital TV Research Lab. and Multimedia Research Lab., LG Electronics, Inc., Seoul, Korea, 1997 – 2003

## **Publications**

### **Journal Articles**

[J3] Kyungwoo Lee, Aviral Shrivastava, Ilya Issenin, Nikil Dutt, and Nalini Venkatasubramanian, "Partially protected caches to reduce failures due to soft errors in multimedia applications", *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, Accepted for publication.

[J2] Seungcheon Kim, Jungae Park, Kyungwoo Lee, and Sangwook Lim, "Home networking digital TV based on LnCP", *IEEE Transactions on Consumer Electronics*, Vol. 48, No. 4, pp. 990-996, Nov. 2002.

[J1] Namkyu Lee, Sungbong Yang, and Kyoungwoo Lee, "Efficient parity placement schemes for tolerating up to two disk failures in disk arrays", *EUROMICRO Journal of Systems Architectures*, Vol. 46,no. 15, pp. 1383-1402, Dec. 2000

### **Conference Papers**

[C7] Kyoungwoo Lee, Aviral Shrivastava, Minyoung Kim, Nikil Dutt, and Nalini Venkatasubramanian, "Mitigating the impact of hardware defects on multimedia applications - A cross-layer approach", *Proceedings of ACM International Conference on Multimedia (ACM MM '08)*, Oct. 2008, Vancouver, Canada.

[C6] Kyoungwoo Lee, Minyoung Kim, Nikil Dutt, and Nalini Venkatasubramanian, "Error exploiting video encoder to extend energy/QoS tradeoffs for mobile embedded systems", *Proceedings of 6th IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES '08)*, Sep. 2008, Milano, Italy.

[C5] Kyoungwoo Lee, Aviral Shrivastava, Nikil Dutt, and Nalini Venkatasubramanian, "Data partitioning techniques for partially protected caches to reduce soft error induced failures", *Proceedings of 6th IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES '08)*, Sep. 2008, Milano, Italy.

[C4] Kyoungwoo Lee, Aviral Shrivastava, Ilya Issenin, Nikil Dutt, and Nalini Venkatasubramanian, "Mitigating soft error failures for multimedia applications by selective data protection", *Proceedings of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '06)*, Oct. 2006, Seoul, Korea.

[C3] Kyoungwoo Lee, Nikil Dutt, and Nalini Venkatasubramanian, "Experimental study on energy consumption of video encryption for Mobile handheld devices", *Proceedings of IEEE International Conference on Multimedia and Expo (ICME '05)*, Poster Session, July 2005, Amsterdam, The Netherlands.

[C2] Shivajit Mohapatra, Radu Cornea, Hyuok Oh, Kyoungwoo Lee, Minyoung Kim, Nikil Dutt, Rajesh Gupta, Alex Nicolau, Sandeep Shukla, and Nalini Venkatasubramanian, "A cross-layer approach for power-performance optimization in distributed mobile systems", *Proceedings of Next Generation Software Program in conjunction with IEEE International Parallel and Distributed Processing Symposium (IPDPS '05)*, April 2005, Denver, CO, USA.

[C1] Seungcheon Kim, Jungae Park, Kyoungwoo Lee, and Sangwook Lim, "Home networking

digital TV based on LnCP”, *Proceedings of International Conference on Consumer Electronics (ICCE '02), Digest of Technical Papers*, June 2002.

**Posters**

[P1] Kyoungwoo Lee, Nikil Dutt, and Nalini Venkatasubramanian, ”A cross-layer, error-aware methodology for reliable design of resource constrained embedded systems”, *11th Annual ACM/SIGDA Ph.D. Forum at Design Automation Conference (DAC '08)*, June 2008, Anaheim, CA, USA.

# Abstract of the Dissertation

Cooperative Cross-Layer Protection for Resource  
Constrained Mobile Multimedia Systems

by

Kyoungwoo Lee

Doctor of Philosophy in Computer Science

University of California, Irvine, 2008

Professor Nikil Dutt, Chair

With rapid advances of technology and wide deployment of wireless communication, mobile embedded systems are becoming popular. However, system failures increase significantly due to increasing complexity of integration and increasing error rates as technology scales, and thus reliability is becoming a critical concern. Incorporating reliability in resource-limited mobile embedded systems poses significant challenges due to high overheads of conventional redundancy techniques in terms of performance, power, cost, etc. For example, error correction codes for cache protection incur more than 22% power overhead, and triple modular redundancy techniques for logic component protection incur 200% cost of area and power without optimization. Using the observation that error-awareness such as error-tolerance, error-resilience, and error-concealment can be exploited to enhance system properties, this thesis proposes a cross-layer methodology for mobile embedded systems with minimal overheads by exploiting error-awareness across system abstraction layers in a cooperative manner. Previously proposed cross-layer methods have focused on power, performance, QoS, and timing issues rather than reliability issues. This thesis



investigates errors and error control schemes across system abstraction layers, and presents error-aware, cross-layer approaches that exploit existing techniques to mitigate the impact of the different classes of errors, and further exploit errors actively for maximal resource savings. This thesis demonstrates the effectiveness of our cooperative, cross-layer methodology in several ways for mobile embedded systems. We have investigated PPC (Partially Protected Caches) architectures that exploit error-tolerance and vulnerability of applications at the application layer to combat soft errors at the hardware layer. We developed EAVE (Error-Aware Video Encoding) that proposes active error exploitation for maximal energy reduction by intentionally dropping video frames at the middleware layer and managing the quality with error-resilience against network errors. Finally, we developed CC-PROTECT (Cooperative, Cross-layer Protection) that jointly orchestrates error detection schemes at the hardware layer, frame dropping and forward error correction at the middleware layer, and error-aware video encoding at the application layer. Our cross-layer approaches significantly reduce resources such as power and area cost for resource-constrained embedded systems, open up an expanded tradeoff space for multi-dimensional constraints, and eventually enable system designers to explore feasible solutions satisfying maximal reliability with minimal overheads of power and performance.

# Chapter 1

## Introduction

### 1.1 Embedded Systems

Embedded systems are defined as computer systems designed for specific functionalities, as opposed to general computer systems. This thesis presents strategies for investigating multi-dimensional tradeoffs for emerging mobile embedded systems in pervasive computing environments, especially for multimedia applications, and their relevant design constraints, with a focus on reliability.

#### 1.1.1 Pervasive Embedded Systems

Embedded systems are everywhere in our everyday lives. We are already living with a lot of embedded systems at home. For example, a recent high definition digital TV set or set-top-box is a typical embedded system, as are other home electronics such as home security systems and appliances that provide intelligence by embedding chip sets to control them. While you are commuting to school or office, you may drive the most sophisticated cars with hundreds of embedded systems to control entertainment boxes or even critical safety functions such as ABS (Anti-lock Braking System). Not only cars but also any other transportation vehicles are highly integrated embedded systems including metro trains and airplanes. Also, you or your vehicles are being watched by a lot of CCTVs (Closed Circuit Televisions), and monitored by sensors. These CCTVs and sensors are deployed in embedded systems for safety, traffic monitoring, privacy, and



Figure 1.1: Pervasive Mobile Embedded Systems

so on. While you are at the office, you may not work any longer without embedded computer systems. You can communicate with your customers over mobile handhelds or smart phones. All-in-one machines including copy, fax, and scanner are essential office equipments and are also embedded systems. When you go shopping, a bar code reader is another example of embedded systems. And payment transactions can be completed via wireless POS (Point of Sale) at shops and restaurants, which is an embedded system for your convenience.

The continual scaling of technology and system integration have allowed the creation of mobile embedded systems, and these are becoming more popular in pervasive computing environments as shown in Figure 1.1. For example, mobile game console, mobile video telephony, and mobile satellite TV are mobile multimedia embedded system. A lot of mobile embedded systems are being deployed around our lives for many applications including entertainment, communication, science, health, business, education, and even military applications. Indeed, embedded systems are already located everywhere around us even though you may not notice whether they

are embedded computer systems. These embedded systems not only affect our personal lives (both physically and mentally) but also change our social behaviors, economic activities, and so on.

### **1.1.2 Mobile Multimedia Embedded Systems and Multi-dimensional Constraints**

Mobile multimedia embedded systems demand several constraints, and it is challenging and essential for system designers to achieve multi-dimension optimizations mainly due to limited resources. Main constraints include:

**Power/Energy.** Since multimedia applications are running on battery-operated mobile devices, low power consumption or/and low energy consumption is a key design concern. A lot of techniques have been investigated to prolong the life of mobile embedded systems, but it is very challenging since complex multimedia processing algorithms demand the huge processing power and the huge amount of data transmissions consume high communication resources.

**Performance.** It is a fundamental constraint not only for mobile embedded systems but also for any other computer system. Especially, due to high complexity of multimedia algorithms and huge amount of data transmission, high performance is highly desired for mobile multimedia embedded systems.

**QoS (Quality of Service).** If users are unsatisfactory with the QoS, low power and high performance embedded systems are useless. Indeed, high QoS is a necessary concern for system designers and manufacturers to design and develop mobile multimedia embedded systems.

**Real-timeliness.** Many mobile multimedia embedded systems have real-time constraints. For example, most video streaming applications and video telephony require soft or firm real-time constraints. Real-timeliness is not only an essential service metric but also a critical property in some particular applications. For instance, mobile multimedia embedded systems can threaten the lives of humans if they miss the deadlines in some critical applications such as military or health-care applications.

**Reliability.** Mobile multimedia embedded systems are increasingly being deployed in strategic and unreachable locations, e.g., in hostile territory and inside the volcano to observe volcanic activity, and are even crucial for saving human life in distress., e.g., video phones. Also, they are more likely to be exposed to harmful sources, causing failures of critical functionality. Thus, reliability is an emerging critical concern for mobile multimedia embedded systems in pervasive

computing environments.

**Cost.** To improve the performance, power consumption, real-timeliness, and reliability, there is a definite requirement to decrease the cost since they have inherent interactions and conflicts among these properties. In particular, it is a main design concern to reduce the number of components and to increase the level of integration for commercializing the mobile multimedia embedded systems in the market.

Mobile multimedia embedded systems demand multiple constraints in part or in total. According to application specifics, mobile multimedia embedded systems require cost-efficient reliability, low power and high performance with minimal QoS loss, real-time energy-efficient high QoS, etc. Thus, system designers must carry out intensive and extensive investigation for multi-dimensional constraint optimizations.

## **1.2 Motivation and Objectives**

### **1.2.1 Errors and Failures**

Due to scaling technology and emerging ubiquitous environments, designers for mobile embedded systems must deal with several types of errors as shown in Figure 1.2. Figure 1.2 presents a layered architecture of mobile embedded systems for multimedia applications (e.g., mobile video conferencing), and several examples of errors at each system abstraction layer. Mobile embedded systems in general are composed of several abstraction layers such as the application layer, the operating system layer, and the hardware layer. At each system abstraction layer, there exist different types of errors while Figure 1.2 shows a few errors as examples.

Since the complexity of embedded systems only increases due to mobile multimedia applications, incorrect design and wrong implementation generate several software bugs and defects at the application layer or the operating system layer. As embedded systems are becoming used in wireless networks, network errors such as packet losses (due to congested routers, faded access points, etc.) in mobile communication are becoming more popular than the previous communication environments in wired only networks, and more importantly they are bursty in nature.

An emerging class of errors are soft errors or transient faults at the hardware layer, that are becoming a critical design concern, especially beyond sub-micron technology.

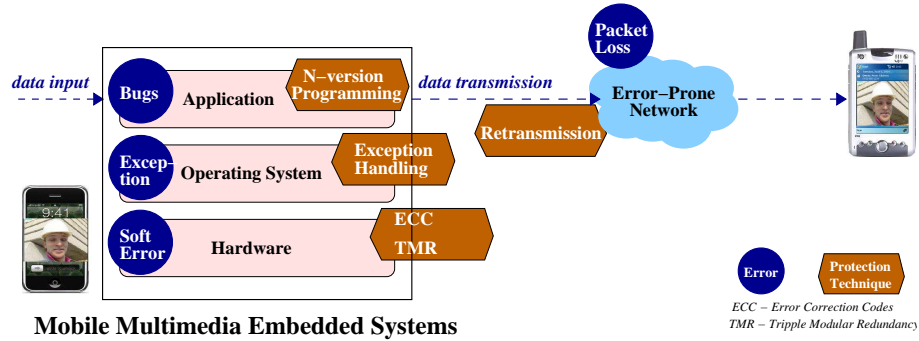


Figure 1.2: Errors and Redundancy Techniques at Each System Abstraction layer

When energetic particles such as alpha particles, neutrons, and protons from packaging material or cosmic rays strike the sensitive area of the silicon device, they generate electron-hole pairs in the wake. The source and diffusion nodes of a transistor can collect these charges,  $Q_{collected}$ . When  $Q_{collected}$  becomes more than some critical value,  $Q_{critical}$ , the state of the logic device, e.g., a Boolean gate, may invert. Since this logic toggle is temporary, the occurrence of such a defect is called a transient fault.

The soft error rate (SER) is related as

$$SER \propto N_{flux} \times CS \times e^{-\left(\frac{Q_{critical}}{Q_s}\right)} \quad (1.1)$$

where  $N_{flux}$  is the intensity of the neutron flux,  $CS$  is the area of the cross section of the node, and  $Q_s$  is the charge collection efficiency [38]. Since  $Q_{critical}$  is proportional to the node capacitance  $C$  and the supply voltage  $V$ , SER has an exponential relationship with the supply voltage as well as the capacitance from Equation (3.1). Thus, with decreasing supply voltage and shrinking feature size, the SER will increase exponentially [38, 124]. In fact, Baumann [8] predicts that the SER in the next generation SRAMs will be up to two orders of magnitude higher. Multiplied by the trend of increasing size of SRAMs in multimedia embedded systems, the SER is becoming an important design concern. Further, the SER is related to where we are running applications in mobile embedded systems. The SER at New York is several times higher than that at the Equator, i.e., the higher latitude, the higher SER. Also, the SER on the airplane at 35,000 feet is several orders of magnitude times higher than the SER at the ground level due to high intensity of the neutron flux as shown in Equation (3.1). As a result, a soft error may occur every less than a

second in mobile embedded systems with 128 MB memory with 65nm technology at flight, and it is a real problem.

Due to existing masking effects, all the errors on mobile embedded systems (such as bugs at the application layer, exceptions at the operating system layer, packet losses at the network layer, and soft errors at the hardware layer as shown in Figure 1.2) are not manifested as failures. Especially, errors on multimedia data itself in mobile multimedia embedded systems may not affect the quality of service significantly nor cause failures. However, these errors on the control data or variables (e.g., conditional variables or loop variables) can cause system failures, e.g., incorrect outputs, application crash, or even the application entering an infinite loop. Mobile embedded systems, which have permeated into almost all aspects of human life, need to be protected from these errors. In particular, mobile multimedia embedded systems will be deployed into hazardous area, remote health-care services, and even military battlefields, where a failure due to errors or faults at any abstraction layer can cause significant fiscal loss or even human life in danger. Thus, these errors or faults must be dealt with when the mobile multimedia embedded systems are used for a critical functionality.

### **1.2.2 Conventional Redundancy Techniques**

Conventional redundancy techniques have been investigated for mobile multimedia embedded systems within the boundary of a couple of system abstraction layers. For example, several software engineering schemes have been studied to reduce the number of bugs at the application layer and at the operating system layer. One of traditional schemes to combat software defects such as bugs is *N-version programming* as shown in Figure 1.2. Many operating systems (OS) provide *exception handlings* to recover OS from errors or exceptions. In networks, there have been a lot of research efforts and one of simple techniques to recover the lost data during transmissions is to *retransmit* the lost data as shown in Figure 1.2. At the hardware layer, many techniques have been proposed to combat temporary faults such as soft errors, and one of the most effective methods to combat errors in the memory system is *an error correction code (ECC) technique* such as a Hamming Code (38, 6).

However, most previous works have focused on issues at a single abstraction layer in a whole system, that generally consists of multiple layers such as hardware, operating system (OS),

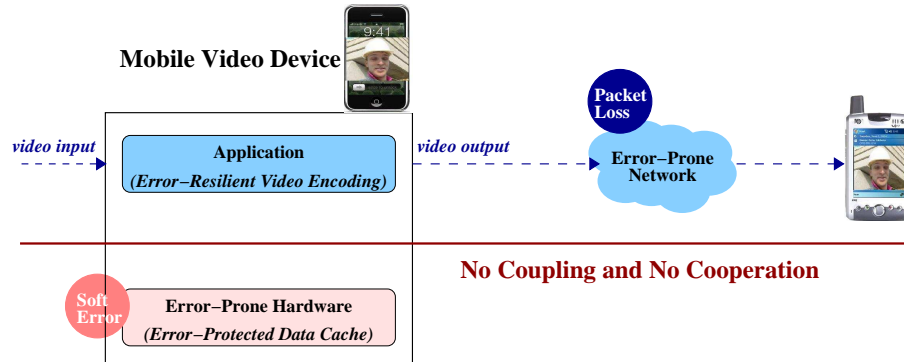


Figure 1.3: No cooperation has been studied to actively exploit existing error-resilient techniques across system abstraction layers.

network, and application layers as shown in Figure 1.2. On one hand, any single variation of the property at a single layer can affect the other properties at other layers in the whole system due to tightly coupled system layers in recent mobile embedded systems. In order to increase reliability of the application, for instance, checkpoints and recovery can be applied at the OS layer; these checkpoints may reduce the chance of voltage scaling for energy gains at the hardware layer, and increase the likelihood of deadline misses of the application, which will be fatal in case of hard real-time embedded systems. On the other hand, isolated schemes at one layer can cause over-protection or under-protection. For example, protecting all data against soft errors in memory systems is an overkill in mobile multimedia embedded systems since we may not need to protect multimedia data that do not cause failures in general. One solution at a single layer for reliability is enough but unnecessary protections are used at the other layers without the global system view, which wastes the limited resources in mobile embedded systems. Furthermore, the mobile computing environments are varying dynamically, where different levels of protection techniques should be considered to provide appropriate reliability without wasting the limited resources such as power and performance.

It has been rarely studied to actively exploit and coordinate existing error-resilient techniques across system abstraction layers as shown in Figure 1.3. Thus, we may lose the opportunities for effective cross-layer methods by coupling resource-efficient techniques. For example, to mitigate the impact of soft errors (at the hardware layer) on the QoS, we can exploit error-resilient video encoding (which was originally developed to combat network errors with perspective of the



QoS) as shown in Figure 1.3, while conventional methods such as an ECC scheme protect soft errors with high overheads in terms of performance, power, and cost.

### 1.2.3 Resource Efficient Reliability

Resource-efficient reliability is essential in mobile multimedia embedded systems mainly due to the limited resource such as battery capacity.

Due to the intensive complexity of processing algorithms and the large amount of data transmission, it is a challenging task to satisfy multiple constraints such as performance, energy consumption and QoS that mobile multimedia systems demand on battery-operated mobile devices. One of the promising approaches to balance these multiple constraints is a cross-layer method. With the global view of the whole system, the cross-layer methods achieve the maximal power reduction and performance gain with the satisfactory QoS. For instance, GRACE [34, 130] proposes a coordinator to reduce the power consumption by exploiting the feature of multimedia applications; DYNAMO [23, 31, 82, 83] presents a proxy-based middleware approach by trading off the video QoS; and recently xTune [53] studies online timing-QoS verification at the proxy server. This thesis studies the cross-layer interactions and potential cooperations among error control schemes to maintain multiple constraints in resource-constrained mobile devices.

## 1.3 Thesis Overview

A cooperative cross-layer approach is our methodology to achieve resource-efficient reliability for mobile multimedia embedded systems as summarized in Figure 1.4.

Our goal is to coordinate reliability approaches among abstraction layers to find the best cross-layered scheme that achieves the maximal reliability with minimal overheads in terms of performance, cost, and power. Specifically, considering error-awareness across layers in a mobile embedded system results in an expanded design space to effectively tradeoff power, performance, QoS, and reliability. Error-awareness also enables system designers to extend the applicability of error features at one layer for protection requirements at the other layer, which causes further improvements in power, performance, and reliability. This thesis presents three cross-layer approaches to accomplish this objective, *resource-efficient reliability for mobile multimedia embed-*

ded systems, which are PPC (Partially Protected Caches), EAVE (Error-Aware Video Encoding), and CC-PROTECT (Cooperative Cross-layer Protection).

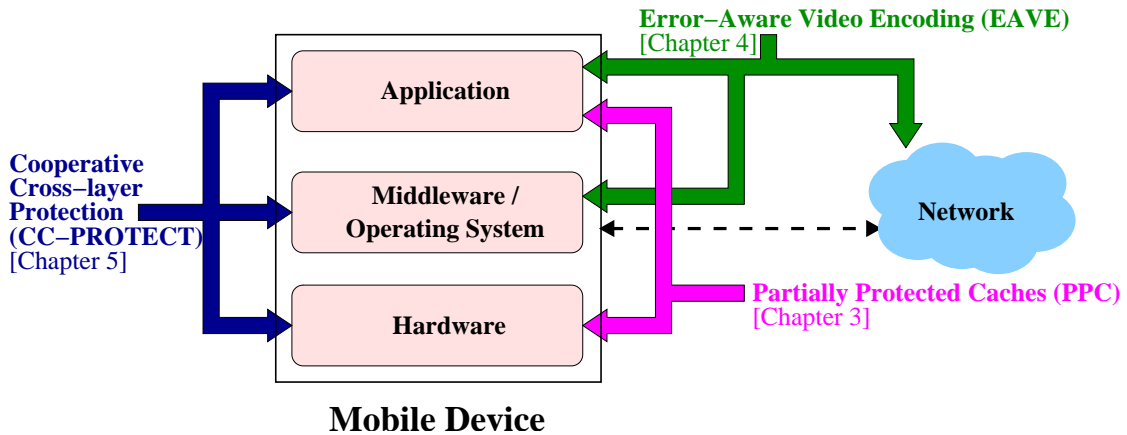


Figure 1.4: Overview of Thesis Proposals – PPC, EAVE, and CC-PROTECT are resource efficient reliability techniques in a cooperative, cross-layer manner.

### 1.3.0.1 PPC (Partially Protected Caches)

ECC-based cache protections are unaware of error-tolerance in applications, and incur high overheads in terms of power and performance. Thus, it is an overkill for multimedia applications since a huge amount of multimedia data is protected unnecessarily with an expensive ECC while hardware defects such as soft errors in multimedia data itself (e.g., an image pixel value) only result in the slight degradation of quality.

This thesis proposes PPC (Partially Protected Caches) and unequal data protection by partitioning data into PPCs. The key idea is that unprotecting failure-non-critical data such as multimedia data itself can significantly reduce the overheads of power and performance while obtaining the comparable reliability to that of a complete data protection (which protects all data). As described in Figure 1.4, the mobile embedded system is aware of error-tolerance inherently existing in multimedia applications at the application layer, and enables the MMU (Memory Management Unit) to map the data into PPCs for protecting data selectively at the hardware layer [65]. This approach can be extended to general applications, where PPC architectures are applied by measuring the vulnerability of data blocks in order to partition data according to the vulnerable time of data residing in data caches [64].

Our unequal protection scheme using PPC architectures demonstrates that a cross-layer approach can improve the performance, energy consumption, cost, and reliability all together at the cost of slight QoS degradation, and opens opportunities that system designers can explore the enlarged tradeoff space among multi-dimensional properties, which are hardly discovered with only fault tolerant schemes isolated at the hardware layer. PPC will be described in detail in Chapter 3.

### **1.3.1 EAVE (Error Aware Video Encoding)**

Error-resilient or error-concealment techniques for multimedia communications have been developed to combat errors induced from the network layer. For example, error-resilient video encoding can adjust the level of error-resilience based on the current or anticipated network status such as a packet loss rate. Fortunately, some error-resilient video encoding works in an energy-efficient way.

This thesis presents an error-aware video encoding (EAVE). The main idea of EAVE is to maximize the features of error-resilient video encoding such as energy-efficiency and error-resilience by injecting errors intentionally at the application layer until the degradation of video quality is maintained by error-resilient features and will be acceptable by end-users [63]. Figure 1.4 shows how EAVE works by correlating approaches and mechanisms across system abstraction layers. The sum of intentional injection rate from the middleware layer and packet loss rate from the network layer becomes an error rate to error-resilient applications so that the error-resilient video encoding techniques adjust the level of error-resilience to generate the error-aware compressed video data dealing with both intentionally injected errors and packet losses.

This cross-layer approach opens a design space where we aggressively tradeoff video quality for improving performance and energy consumption. Further, the main idea, injecting errors intentionally for resource saving, can be applied at each component from the encoder to the decoder in a mobile video telephony to maximize the energy efficiency by exploiting error-awareness very actively, especially for power-constrained mobile embedded systems. EAVE will be discussed in detail in Chapter 4.

### **1.3.2 CC-PROTECT (Cooperative Cross-layer Protection)**

ECC-based cache protections against soft errors or hardware defects are expensive. PPC techniques still incur overheads in terms of power, performance, and cost mainly due to expensive error correction codes installed at the protected cache.

This thesis presents CC-PROTECT (Cooperative Cross-layer Protection), which mitigates hardware defects with cooperative cross-layer protection for resource constrained mobile multimedia embedded systems. This thesis investigates error types, their impacts, and the existing error control schemes across system abstraction layers, and coordinates them to reduce the negative impact of hardware defects in a resource-efficient manner. The main idea of CC-PROTECT is to integrate inexpensive schemes to combat each negative impact resulting from hardware defects. CC-PROTECT protects the component at the hardware layer with inexpensive error detection codes rather than expensive error correction codes, applies middleware-driven approaches for drop and forward recovery, backward error recovery, or hybrid of them for QoS/cost tradeoffs, and exploits an error-resilient video encoding to mitigate the impact of frame drops on the video quality for mobile video encoding systems as shown in Figure 1.4 [69].

CC-PROTECT demonstrates that a cooperative, cross-layer approach can improve the performance, energy consumption, cost, and reliability rather than incurring overheads at the cost of slight QoS degradation, and opens opportunities that system designers can explore the enlarged tradeoff space among multi-dimensional properties.

CC-PROTECT will be presented in detail in Chapter 5.

## **1.4 Thesis Contributions**

### **1.4.1 Effectiveness of Cross-layer Approaches for Reliability**

This thesis presents cooperative approaches considering multiple properties such as performance, energy consumption, cost, reliability and QoS for mobile multimedia embedded systems. Previously, cross-layer approaches have been investigated for mobile embedded systems to tradeoff power, performance, QoS, and timeliness. This thesis expands the cross-layer methodology for reliability-oriented system optimization, and shows the effectiveness of the cross-layer

approach for multi-dimensional optimization.

### **1.4.2 Cost-efficient Reliability**

Holistic schemes aware of errors across layers have been studied and the effectiveness of error-aware, cross-layer methodology has been demonstrated in terms of multi-dimensional metrics such as power, performance, cost, reliability, and QoS [65, 64, 63, 69].

PPC architectures with selective data protection [65] demonstrated about 29% energy saving on an average compared to the protected cache (protecting all data), which is an overkill protection developed at one abstraction layer, especially for multimedia applications. EE-PBPAIR [63], one technique in EAVE, improved the energy consumption by 33% on an average compared to the normal video encoding, which takes into account only compression efficiency at one layer. CC-PROTECT [69] improved the energy consumption and the performance by about 50% with about 1,000 times higher reliability at the cost of minimal QoS degradation, as compared to the composition without any protection and resilience technique for mobile video encoding systems.

### **1.4.3 Expanded Design Space Exploration**

Our cross-layer approach opens up a new design space by actively exploiting error-awareness such as error-tolerance, error-resilience, and error-concealment for maximal energy reduction by trading off a small degradation in the delivered quality of service [65]. Vulnerability-based exploration algorithms have been investigated to expand the applicability of PPC for general applications and for different hardware components other than data caches [64]. Active error exploitation expands the design space significantly [63]. Error-awareness across system abstraction layers opens a new venue where system designers can further coordinate and optimize the system components for multiple constraints [63, 69]. CC-PROTECT presents several exploration algorithms to efficiently find out interesting operation points out of significantly expanded tradeoff space [69].

#### **1.4.4 Extended Applicability of Existing Techniques**

The cooperative cross-layer approaches presented in this thesis significantly extend the applicability of previously proposed techniques. PPC has been proposed based on HPC (Horizontally Protected Caches), which was originally proposed to increase performance, and was extended to improve the energy reduction for embedded systems. PPC is very effective in cache-oriented architecture, such as mobile multimedia embedded systems for multi-dimensional optimization at the system level. For energy saving, error-resilience of applications has been employed by intentionally injecting errors in EAVE. Thus, EAVE expands the error-resilient video encodings to energy-aware and error-aware video encodings while error-resilient video encodings were designed originally to combat network errors such as packet losses. CC-PROTECT also re-discovers and extends the applicability of one approach at one system abstraction layer for the different purpose at the other layer. For example, drop and forward recovery has been applied to increase reliability threatened by soft errors at the hardware layer, which was originally devised to reduce the impact of frame losses due to packet losses at the video decoding at the application layer. For the QoS improvement, error-aware video encoding has been exploited to reduce the impact of soft errors at the hardware layer on the QoS while it was originally developed to compress video data resilient against network errors.

#### **Thesis Organization**

The rest of this thesis is organized as follows: Chapter 2 describes cross-layer approaches in network protocol stacks, and differentiates our methodology from previously proposed cross-layer approaches in mobile embedded systems. Chapter 3 presents partially protected cache (PPC) architectures for soft error mitigation with minimal costs. Chapter 4 presents an active error exploitation with error-resilient video encoding, and proposes error-aware video encodings (EAVE) for energy/QoS tradeoffs. Chapter 5 presents a cooperative, cross-layer protection strategy by exploiting existing error control schemes across system layers for low-cost reliability. In Chapter 6, we conclude this thesis and address the future directions.

## Chapter 2

# Cross-Layer Approach

As the complexity of computer systems and communication architectures has grown dramatically, there was a definite need to develop a layered architecture such as OSI (Open Systems Interconnection) Reference Model as shown in Figure 2.1. Conceptually, a layer is a collection of similar functionalities to send or receive services to the layer above or below it. So a layered architecture divides the overall communication tasks into layers and defines hierarchical services to be provided by the individual layers [99]. Thus, designers and developers can easily implement functions and interfaces between layers according to the specified tasks for each abstraction layer.

However, not only functionality but also resource-efficiency have been considered as an important property for resource-limited mobile computers and wireless communications; therefore a cross-layer approach has recently attracted a significant interest and intensively investigated. Cross-layer approaches integrate and coordinate techniques across communication or system abstraction layers in a cooperative manner mainly for system-level optimization. For example, previous low power techniques for hardware components have been proposed at the hardware layer with the management at the operating system level. They include OS-level scheduling for managing low power processors [122, 75], spin-down policies for disks [39, 21, 22, 71], and wireless network communications [43, 58, 57, 111]. However, cross-layer approaches (adaptively integrating and coordinating these previously proposals) have improved significant power savings and energy reductions [34, 31, 23, 125]. While Kawadia et al. [49] delivered a cautious message for cross-layer designs, cross-layer approaches have been demonstrated as a promising tool to design

not only networking architectures but also mobile embedded systems in the literature.

Cross-layer methods can be exploited at any layering architectures such as network layering and OS layering. We now present network layering approach as a case study of general cross-layer approaches (Section 2.1), describe cross-layer approaches for mobile multimedia embedded systems (Section 2.2), and briefly summarize reliability techniques at each system abstraction layer (Section 2.3).

## 2.1 Case Study: Network Layering

Generic cross-layer methods in the OSI reference model have been widely investigated as promising optimization tools to efficiently reduce the resource consumption, especially transmission energy consumption, in wireless communications. Most efforts have focused on resource optimization at the physical layer by cooperating the feature at the application layer, the routing algorithm at the network layer, or the error/flow controls at the network layer and at the data link layer as shown in Figure 2.1. These cross-layer approaches have been developed not only for wireline networks but also for wireless networks including WiFi network, cellular network, wireless sensor networks, etc.

Many researchers surveyed cross-layer optimizations for wireless networks and presented challenges and open issues for further optimizations. Srivastava and Motani [109] presented a survey of on-going work of cross-layer designs in wireless networks, and suggested new directions and open challenges for cross-layer design. Interestingly, they categorized cross-layer optimization into several approaches such as new interface design between upper layers and lower layers, design coupling, merging of adjacent layers, etc. as shown in Figure 2.1. Also, Su and Lim [112] presented a cross-layer framework composed of an optimization agent and feedback/interaction architectures for wireless sensor networks, and Shakkottai et al. [104] summarized efforts of cross-layer design for wireless networks. Chiang et al. [18] surveyed a recent work of cross-layer optimization for systematic architecture designs in networking. They presented “layering as optimization decomposition”, which provides a mathematical theory to distributively solve generalized network utility maximization (NUM) formulations through decomposed subproblems, a unifying framework for horizontal decomposition to distributed network elements and vertical



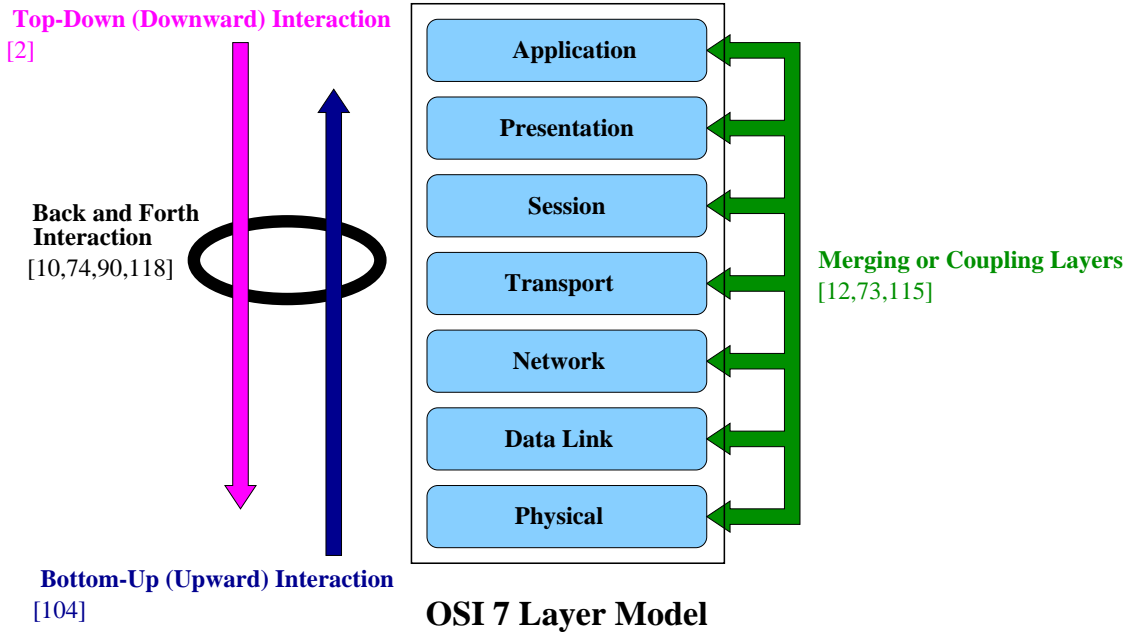


Figure 2.1: ISO OSI 7 Layer Model and Cross-Layer Proposals [109, 112]

decomposition to functional modules, and a top-down approach to design protocol stacks and network architectures. Larzon et al. [2] proposed hints and notifications (HAN) mechanisms to enable inter-layer communications in the Internet architecture. Hints enable real-time applications to transmit the delay requirements and error tolerance to lower layers, and notifications inform upper layers of feedback of link layer actions and current status. On the other hand, Kawadia et al. [49] showed that unbridled cross-layer design can lead to a spaghetti design and difficulties to manage or update due to the lack of modulation. Thus, they revisited conventional network architectures and presented a few general principles for cross-layer design in wireless networks.

One of main contributions in cross-layer optimizations is a medium access control protocol by cooperating information available at upper layers. Butala and Tong [12] proposed a medium access control protocol for CDMA ad hoc networks, which is a cross-layer combination of the scheduling at the medium access control layer and dynamic allocation of channels at the physical layer by means of querying the channel. Similarly, several works [20, 115] have discussed combining cross-layer schemes between medium access control and physical layer design for performance improvements in wireless networks, and especially Tong et al. [115] discussed a

new design paradigm to change the role of the medium access control layer due to new features in the physical layer such as multi-packet reception capability at the same time. Zorzi et al. [139] discussed the issues for cross-layer design of medium access control protocol for multi-radio (MIMO or Multiple-Input Multiple-Output) Ad Hoc networks. Ge et al. [32] considered multicast communications for the rate optimization at the medium access control layer, and investigated joint optimization of the transmission rate and the multicast threshold not only for SISO (Single-Input Single-Output) but also for MIMO.

On the other hand, higher level interactions including the transport layer have been studied as another main stream for cross-layer optimizations for various wireless networks. Bhatia and Kodialam [10] derived a performance guaranteed polynomial time approximation algorithm for jointly solving routing, scheduling, and power control together over multi-hop wireless networks. Lin et al. [74] developed a cross-layer optimization approach such as the opportunistic scheduling problem in access-based single-hop networks such as cellular network, and the joint approach of congestion-control and scheduling problem in multi-hop wireless networks. So their cross-layer optimization cooperates congestion control at the transport layer, routing at the network layer, and scheduling/power control at the MAC/PHY layer. Lin and Shroff [73] also demonstrated the effectiveness of cross-layer design compared to the layered design in terms of the performance in case of imperfect scheduling, and presented a framework for cross-layer congestion control suitable for online and potentially distributed implementation. Chiang [17] proposed a joint optimization between congestion control at the transport layer and power control at the physical layer in wireless multi-hop networks. In particular, he presented a distributed power control algorithm coupling with transmission control protocols to increase both end-to-end throughput and energy efficiency. Ng et al. [90] presented a joint optimization of user data compression at the application layer and channel coding at the physical layer. They presented the optimal power distribution that minimizes the end-to-end expected distortion with each layer successively refining the description in the previous layer.

Interestingly, some cross-layer optimizations considered error controls and flow controls at the higher layers, and combined them for transmission power reductions at the lower layer. Vuran et al. [118] presented a cross-layer methodology to analyze error control schemes with respect to transmission power and end-to-end latency, especially impacts of routing, medium ac-

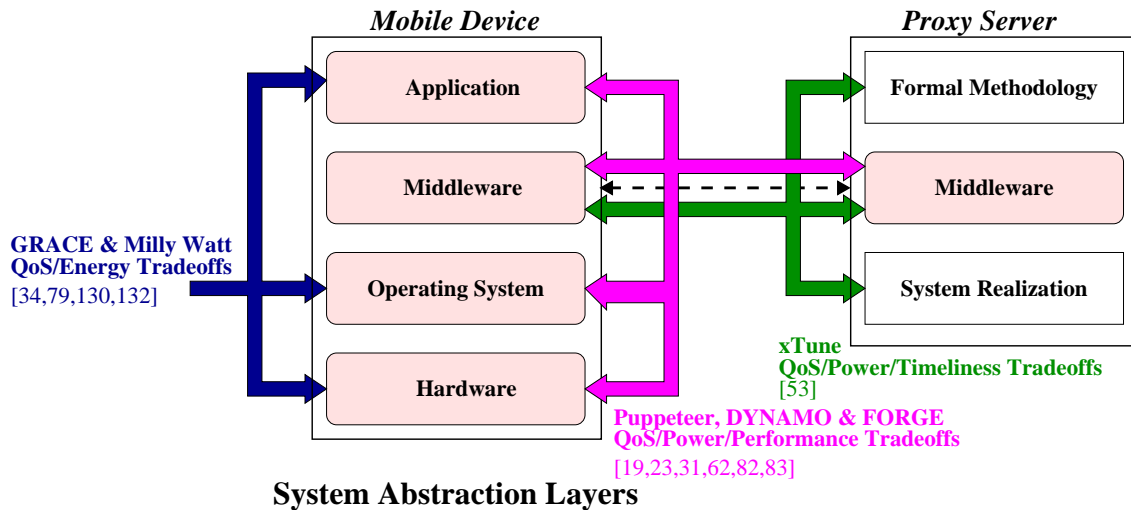


Figure 2.2: System Abstraction Layers for Mobile Embedded Systems and Related Work

cess, and physical levels in wireless sensor networks. For wireless mesh networks, Akyildiz and Wang [3] motivated the cross-layer optimization, and discussed the open problems for cross-layer optimization schemes and algorithms by comparing pros and cons of cross-layer optimization to the layered design schemes.

These efforts have provided the mathematical tools and theoretic backgrounds for cross-layer optimizations in wireless communications including wireless local area networks, cellular networks, wireless sensor networks, and wireless mesh networks, but they have mainly focused on network architecture designs in OSI 7 layer model rather than mobile embedded system designs.

## 2.2 Cross-Layer Approach in Mobile Embedded Systems

This section presents related work on cross-layer methods for mobile multimedia embedded systems, as opposed to schemes isolated within a couple of system abstraction layers. Researchers in general suggest a mobile multimedia embedded system is composed of several system abstraction layers such as the application, the middleware, the operating system, and the hardware layers as shown in Figure 2.2. Efforts have been focused on performance/energy/QoS tradeoffs and optimization (Section 2.2.1), and recently timeliness issues have been discussed together for real-time applications (Section 2.2.2).

### 2.2.1 QoS/Performance/Energy Tradeoffs

Cross-layer optimization techniques have been widely investigated for mobile multimedia embedded systems. Mostly, they minimize the performance and energy overheads while maximizing the QoS for multimedia applications by transporting the available information and coordinating approaches in a cross-layered manner.

Noble et al. [93] proposed the Odyssey prototype for application-aware adaptation by collaborating partnership between the operating system and applications and by identifying agility as a key attribute for system adaptations. Flinn and Satyanarayanan proposed a tool, PowerScope [30], for profiling energy usage of mobile applications. They combined hardware instrumentation with CPU profiling techniques to map power consumption to program structure. Based on PowerScope, Flinn and Satyanarayanan [29] presented energy-aware adaptation for a diversity set of mobile applications. They exploited a collaborative relationship between applications and the operating system to meet user-specified goals for battery duration. Further, their framework on the Odyssey platform enables the operating system to guide runtime adaptation for the better tradeoff between energy consumption and application quality by monitoring energy supply and demand.

The Milly Watt project [79] explored the needs of higher-level (applications and the operating system) involvement for power management techniques. They developed power management functions and a power-based API at the OS level to allow partnership between applications and the system in setting energy policy [25]. Zeng et al. [133] proposed the Currency Model that unifies diverse hardware resources and enables fair allocation of available energy among applications under a single management framework. They also implemented an energy-centric operating system, ECOSystem, that incorporates their system model for explicit energy management with a total system point of view. Lara et al. [62] proposed a component-based middleware architecture, Puppeteer, for mobile computing, and Flinn et al. [28] demonstrated the feasibility of Puppeteer for energy reduction by exploiting well-defined interfaces from applications and modifying their behavior.

Hughes et al. [42] proposed an integrated power management technique by combining architectural adaptation and dynamic voltage (or frequency) scheduling for further energy reduc-

tion in multimedia applications. Yuan and Nahrstedt [128] proposed a middleware framework to reduce the energy consumption and to maintain the resource requirements for multimedia applications. Their middleware framework adjusted the processor speed and power consumption by observing the system properties, and maintained the resource requirement through a power-aware resource reservation mechanism.

The GRACE project [34] presented a cross-layer adaptation framework, GRACE-1 [132], which coordinates all three system abstraction layers (the CPU hardware, OS, and Multimedia application) to achieve a system-level optimization, and balances between multimedia quality and battery energy as shown in Figure 2.2. Yuan et al. [129] proposed an energy-efficient real-time scheduler (GRACE-OS) based on statistical distribution of application cycle demands, and presented a practical voltage scaling algorithm (PDVS) [130] to coordinate adaptation of multimedia applications and CPU speeds for mobile multimedia systems.

Shenoy and Radkov [105] proposed proxy-based techniques for video streaming applications in mobile devices. They employed power-friendly video transformations to tradeoff video quality for energy savings, and suggested an intelligent network streaming strategy for transmission and reception energy savings.

The FORGE project [19, 31] developed not only proxy-based cross-layer optimization but also middleware-driven adaptive coordination with the global system views (both horizontally and vertically) in distributed embedded systems. Mohapatra et al. [83] presented an integrated power management technique considering hardware-level power optimization and middleware-level adaptation to minimize the energy consumption while maintaining user experience of video quality in mobile video applications. In particular, a cross-layer optimization has been addressed for mobile video applications in distributed real-time mobile systems [82]. Mohapatra et al. [84] also presented a cross-layer framework (DYNAMO) and utilized the middleware to perform end-to-end adaptations such as admission control, network shaping, and dynamic video transcoding at the proxy server in distributed embedded systems for QoS/Energy tradeoffs.

Schaar and Shankar [103] presented a conceptual framework of cross-layer optimization for different multimedia applications with different bandwidth intense, delay sensitivities, and loss tolerances, and identified a cooptation-based paradigm and multiple strategies for quality/power consumption tradeoffs in wireless local area networks.

In particular, cross-layer optimizations for video applications have been studied. Schaar et al. [117] proposed a joint cross-layer approach of application-layer packetization and MAC-layer retransmission strategy, and developed on-the-fly adaptive algorithms to improve the video quality under the bandwidth and delay constraint for wireless multimedia transmission.

There have been wide researches on performance/cost/energy/QoS tradeoffs for video communications. Khan et al. [52] analyzed and evaluated the performance gain and the communication cost of cross-layer design for a wireless multi-user video stream application. Khajeh et al. [50] explored a cross-layer design approach to optimize the overall system power consumption by extending cognitive radio platforms and exploiting aggressive Adaptive Voltage Biasing (AVB). They traded off system level errors to reduce the power consumption of WCDMA radio transform while maintains the required quality of service and minimizing the performance degradation.

Researchers have also considered error features of video applications for cross-layer optimizations in wireless communications. Bajic [7] developed cross-layer error control schemes considering joint source rate selection and power management for wireless video multicast. Khajeh et al. [51] recently proposed a cross-layer co-exploration considering algorithm of video encoding (at the application layer) and the power management of a wireless modem (at the physical layer), They exploited the inherent error tolerance of multimedia communications and maximized the energy reduction of wireless modem while maintaining the video quality.

These efforts have focused on coordinating resources across system layers with proxy-based technique, middleware-driven approaches, and OS-level adaptation for power/performance/QoS tradeoffs in mobile multimedia embedded systems. However, they have not taken into account timeliness issue and reliability optimization significantly.

### **2.2.2 QoS/Power/Timing Tradeoffs**

Power management techniques such as power shutdown and voltage/frequency scaling techniques have been investigated intensively. Especially, the inherent tolerance of timeliness in multimedia applications is effectively exploited to maximize resource efficiency such as power reduction.

Qiu et al. [97] proposed a new modeling for optimization techniques between power and QoS management in distributed multimedia systems. They presented the power modeled

multimedia systems based on a generalized stochastic Petri net model, and guaranteed the QoS in the context of timeliness (i.e., the combination of delay and jitter). Hua et al. [41] exploited the tolerance of deadline misses in multimedia applications to maximize the energy reduction of the system. They proposed online and offline voltage scaling techniques by incorporating uncertainties of task execution times while maintaining the quality of service at the user level in terms of completion ratio. Abdelwahed et al. in [1] presented an online control framework for self-managing computer systems. They developed an online controller to manage the desired QoS and to decide the best control action for power management under a time-varying workload.

Recently, Kim et al. [53] proposed a unified framework that allows coordinated interactions among sub-layer optimizers through constraint refinement in a compositional cross layer manner to tune the system parameters. They also presented a compositional cross-layer optimization by coordinating local optimizers and refining constraints in resource-limited real-time distributed systems [55].

These studies have successfully discussed the timing issue in mobile embedded systems, and explored the tradeoff space with performance, energy, QoS, and timeliness for real-time applications. However, the reliability issues and coordinating issues of control schemes across system abstraction layers for further system-level optimization have not been considered clearly.

## **2.3 Reliability across System Layers**

This section discusses reliability issues in mobile embedded systems and classifies errors and their control schemes according to system abstraction layers as shown in Figure 2.3. Failures and protection techniques at the hardware layer will be presented in Section 2.3.1, software failures and protection techniques at the application and middleware layers will be discussed in Section 2.3.2, and failures with their protection techniques will be summarized in Section 2.3.3. Note that errors and error control schemes in this section include limited examples of existing classes and techniques.

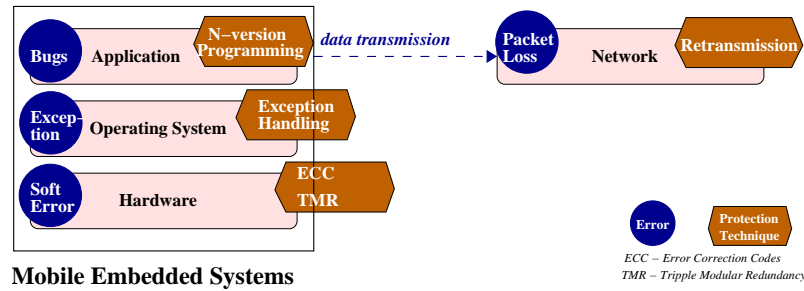


Figure 2.3: Examples of Errors and Redundancy Techniques at Each Abstraction Layer

### 2.3.1 Reliability at the Hardware Layer

Failures at the hardware layer include soft errors, permanent failures, system crashes, etc. These failures result from external radiations such as alpha particles and neutrons, high temperature, battery loss, poor design, and aging. Reliability for hardware components are measured in FIT (Failures in Time), MTTF (Mean Time to Failure), and MTBF (Mean Time between Failures) in general. FIT measures a number of errors in one billion operation hours of a device, MTTF indicates how long it takes to meet a failure and MTBF indicates how long it takes from a failure to the next failure. To recover hardware components from these failures, spatial redundancy and data redundancy techniques have been developed [96]. Spatial redundancy techniques include TMR (Triple Modular Redundancy), duplex, RAID (Redundant Array of Inexpensive Disks) level 1, etc. Data redundancy schemes include ECC (Error Correction Code), EDC (Error Detection Code), RAID level 5, etc.

These hardware failures increase as technology scales and integration increases. For instance, the soft error rate for SRAM (Static Random Access Memory) increases by several orders of magnitude times every technology [8]. Conventional protection techniques are very effective but expensive. For example, an ECC scheme for caches incur up to 95% performance penalty [70] and up to 22% power overhead [95] without any optimization. Recent techniques have focused on the system optimization with minimal overheads and maximal reliability. However, they still incur overheads while our cooperative, cross-layer protection improves the resource efficiency, not incur overheads.



### 2.3.2 Reliability at the Application and Middleware Layers

Examples of failures at the application and middleware layers are wrong outputs, infinite loops, and crashes. The main reasons causing these failures include incomplete specification, poor software design/implementation, programming bugs, and unhandled exceptions. To measure the reliability of software, metrics include the number of bugs per Kilo Lines of Code (KLOC), MTTF, and MTBF. Traditional approaches include spatial redundancy techniques such as N-version programming and N-block recovery, and temporal redundancy techniques such as rollback recovery with checkpoints [96].

As the complexity of a system increases, software errors at the application and middleware layers become dominant. For example, there exists at least several bugs per KLOC when programmers write codes. Further, it is hard to test and debug codes, and it is expensive to apply conventional fault-tolerant techniques. For instance, rollback error recovery with checkpoints is inappropriate for real-time applications since it does not guarantee the completion time of a task.

### 2.3.3 Reliability at the Network Layer

Researches have focused on the network reliability extensively since networks, in particular wireless networks, are unreliable. Briefly, they consider data (frame or packet) losses, deadline misses, node or link failures, and system down. Unreliable network features are main reasons including congested routers, noisy and interfered channels, and even malicious attacks. As quality metrics to measure the reliability of networks, there exist SNR (Signal to Noise Ratio), packet loss rates, deadline miss rates, MTTF, MTBF, and MTTR (Mean Time to Recovery), which indicates how long it takes to recover system from failures.

In network stacks (Figure 2.1), two layers such as data link and transport layers are involved in error controls for hop-to-hop reliability and end-to-end reliability, respectively. Researchers have investigated a lot of techniques, and examples include data redundancy such as CRC (Cyclic Redundancy Check), temporal redundancy such as retransmission, and spatial redundancy such as node replication and multiple radios (e.g., MIMO or Multiple-Input Multiple-Output). Interestingly, there have been several joint techniques to combine multiple approaches across OSI 7 layers for optimal solutions [118, 117].

However, no efforts have been investigated to increase reliability with minimal costs in a cross-layered manner for mobile embedded systems. In the following Chapters, we present our cost-efficient proposals by exploiting existing error control schemes across system abstraction layers in resource-constrained mobile embedded systems, and address explicitly the tradeoffs between reliability and other design constraints such as performance, power, and QoS for system-level optimization.

## Chapter 3

# Partially Protected Caches: Enabling Reliability at the Hardware Layer

### 3.1 Motivation

The increasing incidence and adverse effects of soft errors, or radiation-induced transient faults, pose an overarching challenge in computer system design. Soft error is the phenomenon of temporary change of the state of a logic gate in an integrated circuit under the influence of radiation coming from packaging material or cosmic rays strike on the silicon device. The occurrence of soft errors may have catastrophic consequences for the system: the application may generate incorrect results, try to access protected memory regions, crash, or go into an infinite loop.

Although the phenomenon of soft errors has been known for a long time, only recently have shrinking feature sizes and lowering supply voltages resulted in a significant increase in soft errors, making the presence of soft errors a real design concern. The effects of soft errors become more critical in embedded systems. For example, multimedia embedded systems are used for remote sensing in space, surveillance in hostile territory, monitoring in highly radioactive environments. It should be noted that these embedded systems may be difficult to reach if it is necessary to recover from crash. Thus system failures caused due to soft errors in such systems may be very difficult, if not impossible, to fix.

The occurrence of soft errors is directly proportional to the exposed area of the logic [13]. Since caches are one of the largest area contributors in the processor, they are most vulnerable to soft errors. Previous research has therefore focused on protecting the caches against soft errors. The use of Error Detection Codes (EDCs) like a parity check, and Error Correction Codes (ECCs) like Hamming Codes has been suggested to protect the data in each line of cache. While a single bit EDC can be used to detect single-bit errors, *Single-Error Correction and Double-Error Detection* (SECDED) can correct single-bit errors and also can detect double-bit errors transparently. Since most of the soft errors are single-bit errors (the frequency of double-bit errors is about 2-3 orders of magnitude less than that of single-bit errors), SECDED is a very effective architectural technique to protect caches from soft errors.

However, protecting the caches using SECDED has significant power, performance and area overheads. Every time data from a cache line is read, the ECC check needs to be performed to see if an error occurred in this line. As a result, the error correction logic becomes a part of time-critical path in cache lookup. Previous research indicates that SECDED implementations can increase the cache access time, power consumption, and area by more than 20% [70, 95, 60]. Embedded systems, which have very stringent power and performance requirements, may not be able to afford such high overheads. Thus there is a critical need for effective, yet low-overhead architectural techniques to combat soft errors.

This chapter proposes the effectiveness of the cross-layer approach to design a hardware/software joint solution to mitigate the impact of soft errors for cache-oriented embedded systems. Our approach to reduce such overheads is based on the observation that in multimedia applications, *not all data is equally failure critical*. The image data, or audio data, is not as critical for failure as the loop variables or the stack pointer. While the occurrence of a soft error in an image pixel may only result in a slight degradation in the image quality, a soft error in the loop variable may result in a segmentation fault. In such a case, we say that the image pixel is a *failure non-critical* (FNC) data item, while the loop variable is a *failure critical* (FC) data item. While it is important to keep all the failure critical data in a soft error protected cache, the failure non-critical data can be kept in cache that is not protected from soft errors.

To exploit the difference in the failure-criticality of the data in multimedia applications, we propose a novel architecture - Partially Protected Cache (PPC). A PPC architecture maintains

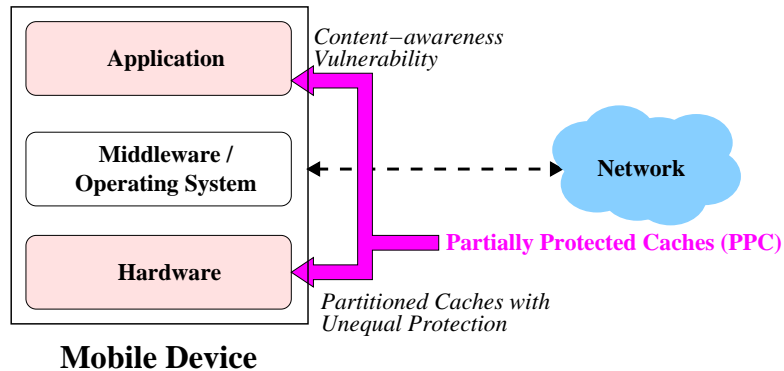


Figure 3.1: Overview of our proposal, PPC and unequal protection in a cross-layered manner

two caches at the same level of memory hierarchy. One of the caches is protected from soft errors, while the other one is unprotected. By mapping the failure critical data into the protected cache, and mapping the failure non-critical data into the unprotected cache, the failure rate of applications can be drastically improved. Note that this improvement in failure rate is obtained mostly at the cost of QoS with very small impact on power and performance.

As shown in Figure 3.1, PPC provides an unequal protection by exploiting the feature of applications, e.g., error-tolerance of multimedia data, for resource-constrained embedded systems. While an obvious partitioning technique exists for multimedia applications, there is no known partitioning techniques for data and code in general applications. To efficiently find out data and code partitions for PPC architectures in general applications, this chapter also presents vulnerability-based page partitioning techniques.

In the rest of this chapter, we focus on developing a microarchitectural solution (PPC) and page partitioning techniques for resource-constrained embedded systems. Thus, our PPC solution with unequal protection exploits the content-awareness (e.g., the error-tolerance of multimedia data) and the time-awareness (e.g., the vulnerability of data and code and its relationship with SER) in a cross-layered manner to combat soft errors with minimal costs, as shown in Figure 3.1.

## 3.2 Related Work

### 3.2.1 Soft Errors

Soft errors, i.e., transient faults, bit-flips, or single-event upsets (SEU), are caused primarily by external radiations in microelectronic circuits, and have been investigated extensively since late 1970's.

Figure 3.2 [77] illustrates the mechanism of a soft error event in a CMOS device. When energetic particles such as alpha particles, neutrons and protons from packaging material or cosmic rays strike on the silicon device, they generate electron-hole pairs in the wake. The source and diffusion nodes of a transistor can collect these charges,  $Q_{collected}$ . When  $Q_{collected}$  becomes more than some critical value,  $Q_{critical}$ , the state of the logic device, e.g., a Boolean gate may invert. Since this logic toggle is temporary, the occurrence of such a defect is called a transient or soft error.

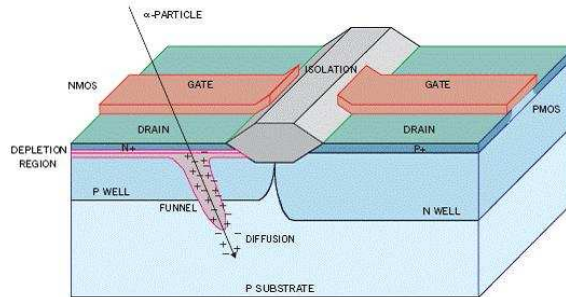


Figure 3.2: External radiation may induce soft error

### 3.2.2 Soft Error Rate and Vulnerability

The soft error rate (SER) in Figure 3.2 is related as

$$SER \propto N_{flux} \times CS \times e^{-\left(\frac{Q_{critical}}{Q_s}\right)} \quad (3.1)$$

where  $N_{flux}$  is the intensity of the neutron flux,  $CS$  is the area of the cross section of the node, and  $Q_s$  is the charge collection efficiency [38]. Since  $Q_{critical}$  is proportional to the node capacitance  $C$  and the supply voltage  $V$ , SER has an exponential relationship with the supply voltage as well as

the capacitance from Equation (3.1). Thus, with decreasing supply voltage and shrinking feature size, the rate of soft errors will increase exponentially [38, 124]. In fact, Baumann [8] predicts that the SER in the next generation SRAMs will be up to two orders of magnitude higher while the SER of DRAMs has been saturated. Multiplied by the trend of increasing size of SRAMs in multimedia embedded systems, the SER is becoming an important design concern.

On the other hand, the unit of SER, FIT (Failures in Time), is the number of soft errors per Mbit for one billion-operation hours. Thus, the SER is linearly proportional to the cache size and the execution time apparently. However, the possibility of a fault (i.e., upset) to effect an error (i.e., soft error) is not directly proportional to the cache size and the execution time since all upsets do not propagate errors. Therefore, there is a definite need for an accurate metric to estimate soft errors.

Substantial work has concentrated on estimating the soft error rate, and the corresponding failure rate. Architectural Vulnerability Factors (AVF) [88] was defined as the probability that a fault in a particular structure will result in a visible error in a final output. However this factor is constant for a given structure, and is not application dependent. This methodology has been widely exploited [80], and also extended to present the vulnerability of the cache systems [6, 5]. In particular, Asadi et al. presented the critical time of the critical word, the residency time of the word data in caches, and examined the vulnerability of the cache components and the effects of cache policies such as flushing, write-thru and refreshing. However, they did not capture the byte-level residency time, and ignored the effects of the write operation in a word on the other words in the same line at eviction. Similarly, the temporal vulnerability factor (TVF) [119] and the cache vulnerability factor (CVF) [135] have been proposed as metrics to estimate the vulnerability of the caches to soft errors. However, they failed to present the actual relationship of the vulnerability factor and the failures of applications.

### **3.2.3 Soft Error Protection Techniques**

Solutions to combat the challenge of soft errors have been proposed at various levels of design abstraction from process technology to architectural solutions. We briefly review related work and position our research with respect to these efforts.

### 3.2.3.1 Packaging and Process Technology Solutions

Radioactive substances such as alpha particles emitted by packaging and wafer processing materials are one of the major sources of radiations that cause soft errors in semiconductors. Thus, advances in process technology such as purification of packaging materials, radiation hardening, and elimination of Boron-10 ( $B^{10}$ ) impurities are expected to mitigate the soft errors [9]. However, the effects of interaction between high energetic cosmic particles (e.g., neutrons) and radioactive materials cannot be prevented completely [77].

Process technology solutions such as SOI (Silicon On-Insulator) processes [89, 102] have been proposed. In order to mitigate the soft errors, they extend the depletion region or raise the capacitance, which increases the critical charge of semiconducting devices. However, process engineering technology may require the cost of additional process complexity, the loss of manufacturability, and extra substrate cost [8].

### 3.2.3.2 Processor Architecture Solutions

**Solutions for Combinational Logic.** Logic elements were considered more robust against soft errors than memory elements mainly due to the masking effects. However, many researchers predict that the logic soft errors will become one of main contributions to the system unreliability [107, 8, 92]. The simplest and most effective way to reduce failures due to soft errors in combinational logic is Triple Modular Redundancy (TMR) [96], which typically uses three functionally equivalent replicas of a logic circuit and a majority voter. But the overheads of hardware and power for conventional TMR exceed 200% [92]. Duplex redundancy [81, 92] is also available but it requires more than 100% area and power overheads without any optimization techniques. In order to reduce the high overheads in conventional redundancy techniques, Mohanram et al. [81] presented a partial error masking by duplicating the most sensitive and critical nodes in a logic circuit based on the asymmetric susceptibility of nodes to soft errors. Nieuwland et al. [92] proposed a structural approach analyzing the soft error rate sensitivity of combinational logic to identify the critical components at circuits.

**Solutions for Sequential Logic.** Temporal redundancy is another main approach that has been used to combat soft errors in circuits. In order to detect soft errors, Nicolaidis [91] applied



fine time-grain redundancy within the clock cycle greater than the duration of transient faults by using the temporal nature of soft errors. Similarly, Anghel et al. [4] exploited the temporal nature to detect timing errors and soft errors by means of time redundancy. Krishnamohan et al. [59] proposed the time redundancy methodology by using the timing slack available in the propagation path from the input to the output in CMOS circuits. A Razor [26] flip-flop was presented to detect transient errors by sampling pipeline stage values with a fast clock and with a time-borrowing delayed clock.

**Solutions for Memories.** By far, reducing soft errors in memories has been the most extensive research topic. Error detection and correction codes (EDC and ECC) have been widely investigated and implemented as the most effective scheme to detect and correct soft errors in memory systems. However, an ECC system consists of an encoding block as well as a decoding block responsible for detection and correction, and of extra bits storing parity values. Thus, ECC-based techniques consume extra energy and incur performance delay as well as additional area cost [96, 95, 70, 60], and are therefore not suitable for caches. Thus, only a few processors such as the Intel Itanium processor [98] protect L2 and L3 caches with ECC [110], but we are not aware of any processor employing ECC-based protection mechanism on L1 cache. This is mainly due to high overheads of ECC implementation [56, 85, 138]. Zhang et al. [137] proposed in-cache replication where the dead cache block space is recycled to hold replicas of the active cache block. Also, Zhang [136] presented replication cache where a small fully associative cache is added to keep the replica of every write to the L1 data cache. However, these techniques incur overheads to maintain replicas. A cache scrubbing technique [87] has been proposed, which can fix all single-bit errors periodically and prevent potential double-bit errors. Li et al. [72] evaluated the drowsy cache and the decay cache exploiting voltage scaling and shut-down schemes, respectively, in order to efficiently decrease the power leakage. They also proposed an adaptive error correcting scheme to different cache data blocks, which can save energy consumption by protecting clean data less than dirty data blocks. Kim [56] proposed the combined approach of parity and ECC codes to generate the reliable cache system in an area-efficient way. However, they all exploit expensive error correcting codes in order to protect all the data unnecessarily.

### **Partially Protected Cache Architecture**

We have proposed the PPC architecture and demonstrated the effectiveness in reducing

the failure rate with minimal power and performance overheads [65]. However, the effectiveness of PPCs has been limited only on multimedia applications, and there is no known approach to use PPCs for both data and instruction caches in general applications.

### 3.2.4 Software Solutions

Software-only techniques have been studied to protect data and code from soft errors. Both software and hardware techniques have their own advantages and disadvantages in combating the impacts of soft errors. For example, hardware techniques increase the resource cost with high effectiveness to detect and even correct errors while software solutions mostly do not incur hardware costs with minimal coverage such as only error detection.

Reis et al. [101] presented the software-implemented fault tolerance (SWIFT) for soft error detection by exploiting unused resources and enhancing control-flow checking. Also, Lucetti et. al [76] proposed software mechanisms to tolerate soft errors by leveraging virtual machine and memory sharing techniques. However, they are limited only to detecting errors, and must be used in conjunction with recovery techniques. Through the user-specified annotations, the compiler can separate and map data elements in programs to reliable domain which has protection techniques against soft errors, and to unreliable domain without protection [14]. But it requires the annotation for important data by user specification.

Soft error detection in software is extremely expensive in terms of delay, while it can be done without much overhead in hardware. In contrast, since the soft error rate is very low (as compared to processor clock cycle), soft error correction is efficient in software while it incurs too much overhead in hardware. Consequently, a combined approach that achieves the best of both hardware and software solutions will be very efficient. However, there is no hardware-software hybrid approach for soft error mitigation in resource-constrained embedded systems.

The PPC architecture with software page partitions is the promising one as a joint solution of hardware-software techniques. The compiler separates the failure critical and failure non-critical data and maps each of them into the two caches in a PPC for the selective data protection technique in multimedia applications [65]. However, there is no obvious partitioning technique for general applications, and this thesis proposes a general page partitioning technique not only for data PPCs but also for instruction PPCs.

### 3.3 Architecture of Partially Protected Caches (PPC)

To provide different levels of protection against soft errors, we propose a novel cache architecture, Partially Protected Cache (PPC). Our concept of Partially Protected Cache is derived from the concept of Horizontally Partitioned Caches (HPC) [33]. HPC is a promising technique in which the processor has multiple (typically two) caches at the same level of memory hierarchy, and partitioning the application data and code wisely between the two caches can improve both performance and energy consumption [33, 108]. Similarly, PPC architectures will have multiple caches at the same level of memory hierarchy, varying in the level of soft error protection they provide. In particular, we consider two caches at the L1 level, named the protected cache and the unprotected cache as shown in Figure 3.3. The protection against soft errors in the protected cache can be provided by any of the existing techniques, e.g., increased transistor size, increased supply voltage, SEC-DED, etc. In this thesis, we consider that the protected cache has SEC-DED to correct one bit error and detect two bit errors.

The memory is mapped to the two caches at a page level of granularity. Each page has a Cache Mapping Attribute or CMA. CMA defines which cache the page is mapped to. When a new page is requested in the cache, it comes into the cache defined by the CMA. CMA is stored in the Translation Look-aside Buffer (TLB) along with the address mapping. When the processor requests any cache data, first the TLB lookup is performed to see if the page is present in the cache, and if yes, to figure out which of the two caches is selected. Therefore, only one cache lookup is performed per access. Note that our approach for data partitioning into the two caches does not increase the number of pages, nor the number of TLB misses. Every time data is written into the cache, the data has to be encoded, and every time it is read from the cache, the data needs to be decoded and a check needs to be performed for occurrence of soft errors. Thus, the SEC-DED decoder becomes a part of timing critical path, and has power and performance overheads.

In order to minimize the performance impact of the PPC architecture, the protected cache should be small, so that the total penalty of the protected cache and the SEC-DED implementation is less than or equal to that of the unprotected cache. However, since the protected cache is now smaller, it is important to map data very carefully into this cache. Mapping too much data into the protected cache can result in frequent misses, and therefore degrade performance.

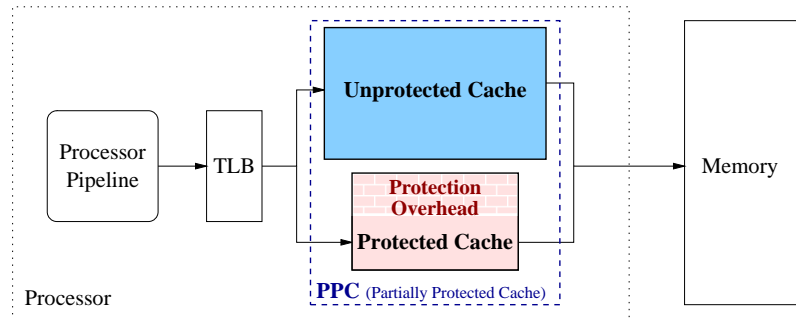


Figure 3.3: PPC (Partially Protected Caches) - one protected and the other unprotected at the same level of memory hierarchy

## 3.4 Page Partitioning Techniques

### 3.4.1 Application Data Partitioning in Multimedia Applications

To compare the susceptibility of application data on soft errors, we devised a simple experiment. The data that the application accesses is divided into pages. We injected soft errors randomly in only one page, and simulated the application several times to estimate the failure rate. Soft errors were injected with a constant probability per line, and per unit time, only in the lines of the page that are in the cache. Figure 3.4 shows soft errors in some pages are much more likely to cause an application failure than soft errors in other pages. In fact, for an image processing benchmark - *susan edges* from the MiBench suite [36], soft errors in only some of the pages (9 out of 83) cause an application failure. Soft errors in most of the other pages do not result in a failure. The degradation in quality typically shows up in the form of white/black pixels in the output image. A simple analysis reveals that the pages that do not cause failures are the image data, and the pages that contain stack variables, and other program variables cause failures. Existing solutions that protect the whole cache using ECC is an overkill for these multimedia data pages, especially in situations where a little loss in quality of service can be tolerated. An ideal solution would provide protection to only the pages that may cause application failures, and reduce the overheads by not protecting data that may not cause application failures.

To exploit the PPC architecture, the compiler has to categorize and partition the application data into the protected and the unprotected cache. In general, a detailed analysis for each variable is needed to be able to partition the application; fortunately the characteristics of mul-

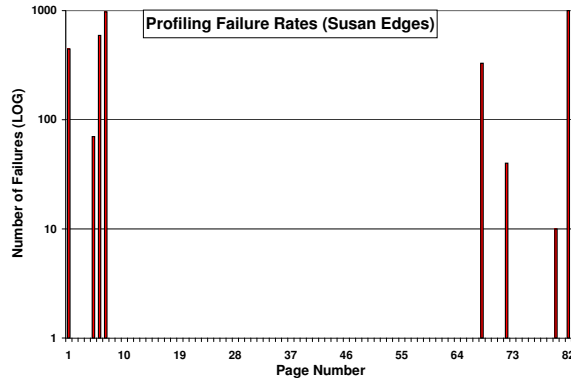


Figure 3.4: Failure rate distribution (benchmark : *susan edges*) - failures are reported in occurrences of soft errors at only 9 pages out of 83

multimedia applications simplify this analysis. In multimedia applications, while a soft error in an image pixel may only cause minor distortion in the image, or negligible loss in QoS, a soft error in the loop control variable may result in memory segment violation, or a failure. Other examples of failures caused by soft errors include system crash and the infinite execution of a loop. For multimedia applications, we define the multimedia data as failure non-critical (FNC), and all the rest of the data as failure critical (FC).

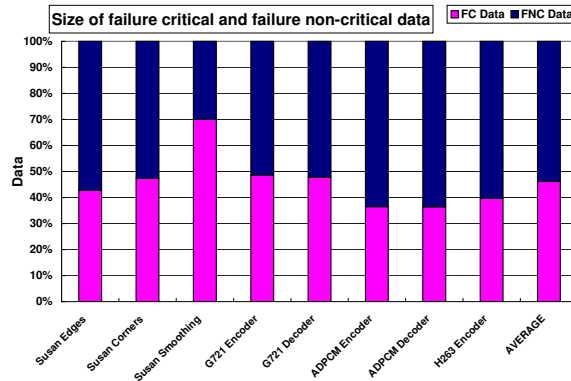


Figure 3.5: Size of failure critical and failure non-critical data in applications

Figure 3.5 plots the percentage of failure critical and failure non-critical data in the various multimedia benchmarks, as found by our method. We have observed that even this simple strategy can mark between 30% to 63% of data as failure non-critical. A better data analysis technique can discover additional failure non-critical data, and therefore will improve the effectiveness

of our technique. However, even the simple technique of finding the failure critical data is quite effective. Also note that it is very easy for the designer to manually identify the multimedia data, since it is typically present in large arrays.

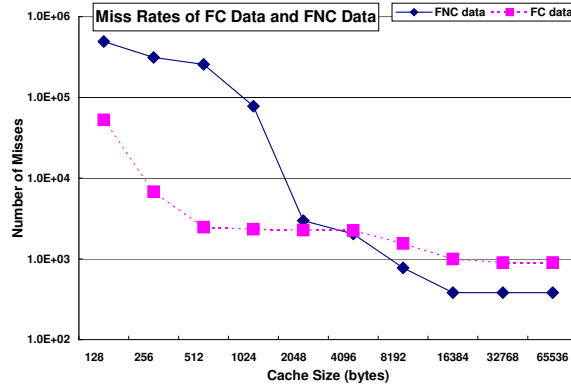


Figure 3.6: Cache miss rates of failure critical and failure non-critical data (benchmark - *susan smoothing*)

The other concern with mapping data to the small protected cache in PPC might be the possible negative impact on performance. Figure 3.6 plots the cache miss rates of the failure critical and failure non-critical data for various cache sizes. We observe that the slope of the miss rate of the failure critical data is less than that of the failure non-critical data. This implies that the size of the protected cache can be reduced without much performance penalty. The reason behind this observation is that the failure critical data that we have marked comprises of the local variables, function stack, etc. which have much better cache behavior than that of the multimedia data. As a result, we can achieve low failure rates without significant power and performance overheads.

### 3.4.2 Page Partitioning in General

#### 3.4.2.1 Vulnerability: A Metric for Failure Rate

To choose pages to be mapped to the protected cache, we need a metric to quantitatively compare page partitions in terms of susceptibility to soft errors. We use the concept of vulnerability [5, 88, 119, 135], to partition the data into the protected and unprotected caches in a PPC. We observe that if an error is injected in a variable that will not be used, the error does not matter.

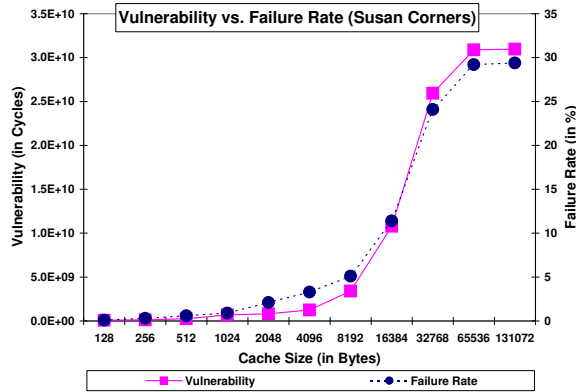


Figure 3.7: Vulnerability and Failure Rate: vulnerability is a good metric for estimating failure rate

However, if the erroneous value will be used in the future, then it will result in a failure. Thus a data is defined to be **vulnerable** for the time it is in the unprotected cache until it is eventually read by the processor or written back to the memory. The vulnerability of an application is just the summation of the individual data vulnerability, which is measured in cycles to present the vulnerable time of this data.

To validate our idea using vulnerability as a failure rate metric, we simulated the *susan corners* benchmark from MiBench suite on a modified *sim-outorder* simulator from SimpleScalar to model HP-iPAQ like system for various L1 cache sizes. Figure 3.7 plots the *vulnerability* and the *failure rate* obtained by simulations. To estimate the failure rate, we injected soft errors on data caches for each execution of the benchmark, counted the number of failures out of a thousand executions, and calculated in a percentage by multiplying 100 with the number of failures divided by the number of runs. Each execution is defined as a success if it ends and returns the correct output. Otherwise, it is a failure. Figure 3.7 shows that the shape of the *vulnerability* closely matches the failure rate curve. Other applications also show similar trends. On average, the error in predicting the failure rate using *vulnerability* metric is less than 5%. In this chapter, we use *vulnerability* as the metric to estimate the failure rate, and perform automated design space exploration to decide the page partitioning between the two caches of a PPC. Reducing vulnerability can be contrary to performance improvement. For example, to reduce the vulnerability of data, data should not remain in the cache for long. It is better to evict and reload the reused data to reduce the vulnerability, but this may degrade performance. Therefore, there is a fundamental trade-off

between performance improvement and vulnerability reduction.

### 3.4.2.2 Page Partitioning: DPExplore

---

```

DPExplore(rPenalty, eWidth, pCount)
01: pageMap0 = 0...0
02: runtime, power, vulnerability = simulate(pageMap0)
03: config0 = (pageMap0, runtime, power, vulnerability)
04: for (k = 0; k < eWidth; k++)
05:   bestConfigs.insert(config0)
06: endFor
07: for (;)
08:   newBestConfigs = bestConfigs
09:   for (i = 0; i < eWidth; i++)
10:     for (j = 0; j < pCount; j++)
11:       testConfig = addPage(newBestConfigs[i], j)
12:       runtime, power, vulnerability = simulate(testConfig.pageMap)
13:       if (runtime < config0.runtime ×  $\frac{100+rPenalty}{100}$ )
14:         if (vulnerability < newBestConfigs[0].vulnerability)
15:           newBestConfigs.insert(testConfig, runtime, power, vulnerability)
16:         endIf
17:       endIf
18:     endFor
19:   endFor
20: endFor
21: for (i = newBestConfigs.length(); i > eWidth; i--)
22:   newBestConfigs.delete[i - 1]
23: endFor
24: if (newBestConfigs[0].vulnerability < bestConfigs[0].vulnerability)
25:   bestConfigs = newBestConfigs
26: else break;
27: endIf
28: endFor

```

---

Figure 3.8: DPExplore: an exploration algorithm for data partitioning

Figure 3.8 outlines our DPExplore partitioning algorithm, which starts from the case when no page is mapped to the protected cache. In each step, pages are moved from the unprotected to the protected cache, to minimize the vulnerability under the runtime penalty. Our page partitioning algorithm takes two parameters: (i) allowable runtime penalty (*rPenalty*), and (ii) exploration width (*eWidth*), i.e., how many partitions are maintained as best configurations for the whole exploration. DPExplore uses *pCount*, the number of pages in a benchmark, and searches for page mappings that will suffer no more than the specified runtime penalty, while trying to minimize the vulnerability. DPExplore maintains a set of best page mappings found so far (Line 05) in *bestConfigs*, sorted in increasing order of vulnerabilities. After initialization, the algorithm goes into a forever loop in Line 07. It takes each existing best solution and tries to improve it by mapping a page to the protected cache (Lines 11-12). If the new page mapping is better than the



worst solution in the *newBestConfigs*, then the new page mapping is saved in the list. The loop in Lines 09-20 is one step of exploration. After each step, the new set of page mappings is trimmed down to exploration width (Lines 21-23). The termination criterion of the exploration is when an exploration step cannot find any better page mapping. In other words, no page can be mapped to the protected cache to improve vulnerability (Lines 24, 26) under the runtime penalty. Otherwise, the global collection of the best page mappings are updated (Line 25). The complexity of our DP-Explore is  $O(N!Wm)$ , where  $N$  is the number of pages to be explored,  $N!$  is  $N \times (N - 1) \times \dots \times 1$ ,  $W$  is the exploration width, and  $O(m)$  is the complexity of a simulation to evaluate a page partition.

Note that our exploration technique is a profile-based approach, which works well if the page mapping of application codes and input data does not change. Our proposal, DPExplore, is very effective for such applications.

### 3.5 Effectiveness of PPC

#### 3.5.1 Experimental Framework

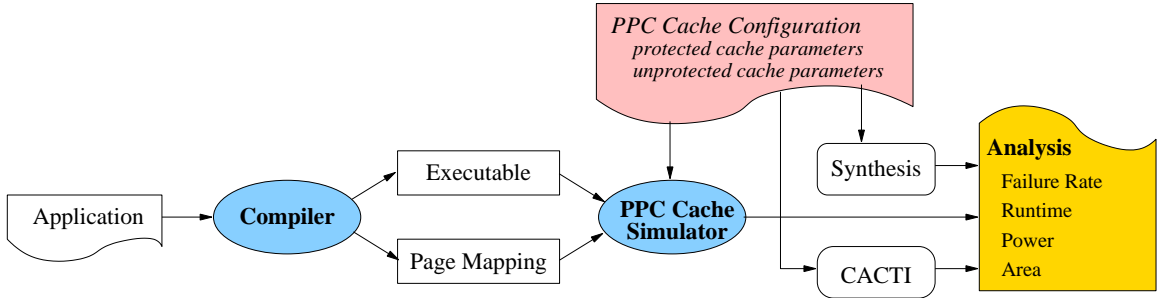


Figure 3.9: Experimental Framework

To demonstrate the effectiveness of PPC architecture, we have developed a compiler-simulator-analyzer framework as shown in Figure 3.9. A compiler generates a page mapping list and an executable, a simulator (modified *sim-cache* and *sim-outorder* simulators from SimpleScalar [11]) runs an executable by mapping pages according to cache configurations, and the outputs are analyzed in terms of performance, power, failure rate (or vulnerability), and QoS. Simulation frameworks are detailed in several publications [65, 64].

### 3.5.2 Experimental Results

#### 3.5.2.1 Effectiveness of PPC for Mobile Multimedia Applications

Similar to caches in the Intel XScale architecture, the Unsafe cache configuration consists of a 32 KB unprotected cache, the Safe cache configuration consists of a 32 KB protected cache, and the PPC configuration consists of a 32 KB unprotected cache and a 2 KB protected cache.

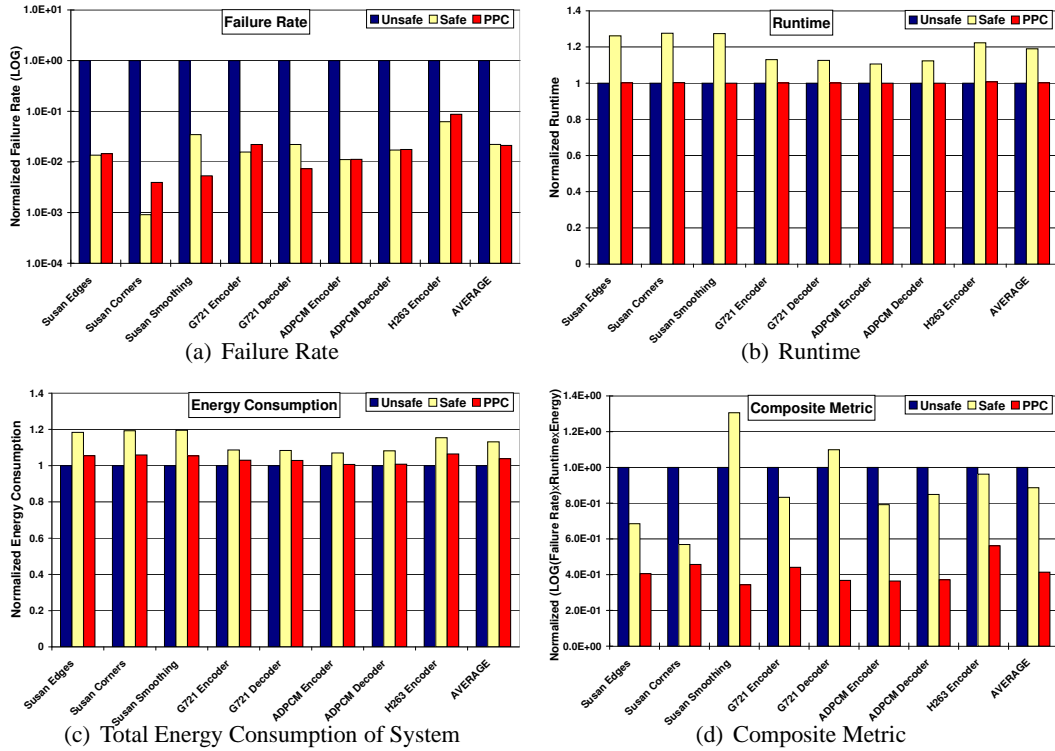


Figure 3.10: Effectiveness of our PPC architectures - PPC achieves minimal failure rates with minimal energy and performance overheads

Figure 5.8(a) shows that the PPC configuration achieves failure rates close to those of the Safe cache configuration, and both of these configurations achieve failure rates about  $47\times$  less than that of the Unsafe cache configuration on an average. Figure 5.8(a) plots the failure rates achieved by the three cache configurations on a logarithmic scale, and they are normalized to the failure rate of the Unsafe cache configuration. In both the Safe and PPC configurations, failures occur only due to double-bit soft errors since all the single-bit errors are corrected by SEC-DED

implementation in protected caches while both single-bit soft errors and double-bit soft errors cause failures in the Unsafe configuration. Figure 5.8(a) shows that in some benchmarks (e.g., *Susan Smoothing*) PPC configuration results in lower failure rate than the Safe cache configuration and in some benchmarks (e.g., *Susan Corners*) the Safe cache configuration is better. However, in most benchmarks PPC configurations provide failure rates close to those of Safe cache configurations, and much lower than those of Unsafe cache configurations.

Figure 3.10(b) shows that PPC configuration achieves the performance close to that of the Unsafe configuration, and incurs only less than 1% performance overhead than the Unsafe configuration on an average while the Safe configuration incurs about 19% performance overhead mainly because of cache access time penalty. Figure 3.10(b) plots the runtime for the three cache configurations. The runtimes are normalized to the runtime of the Unsafe cache configuration. The plot shows that as compared to the previously proposed Safe cache configuration, the PPC configuration has on an average 16% performance improvement.

Figure 5.8(c) shows that PPC configuration consumes less system energy than the Safe configuration. Figure 5.8(c) plots the system energy consumption of the three cache configurations normalized to that of the Unsafe cache configuration. The plots show that as compared to the Safe cache configuration, the PPC configuration consumes 8% less energy on an average. As compared to the Unsafe cache configuration, the PPC configuration consumes about 3% more energy while the Safe configuration consumes 13% more energy on an average mainly because of unnecessary protection for multimedia data, which is saved by PPC approach. Note that the energy consumption indicates the energy consumed by the whole system including processor, data cache, memory, and off-chip buses. If we consider only energy consumption of the memory subsystem, PPC configuration is more effective since the processor energy consumptions for each cache configuration are the same. For example, PPC cache configuration consumes the memory subsystem energy 20% less than the Safe configuration.

To simplify the multi-dimensional comparison of various metrics (failure rate, energy, runtime), we define a composite quality metric ( $CM_{cfg}$ ) for each cache configuration,  $cfg$ , as  $CM_{cfg} = F_{cfg} \times R_{cfg} \times E_{cfg}$ , where  $F_{cfg}$  is the logarithmic failure rate,  $R_{cfg}$  is the runtime, and  $E_{cfg}$  is the energy consumption for a cache configuration  $cfg$ . The lower the composite metric, the better the configuration. Figure 3.10(d) shows the composite metric for all the three cache

configurations for each benchmark. The plot clearly shows that the PPC configuration is a superior design choice. The PPC configuration is  $40\times$  better than the Unsafe cache configuration, and  $2\times$  better than the Safe cache configuration in terms of a composite metric.

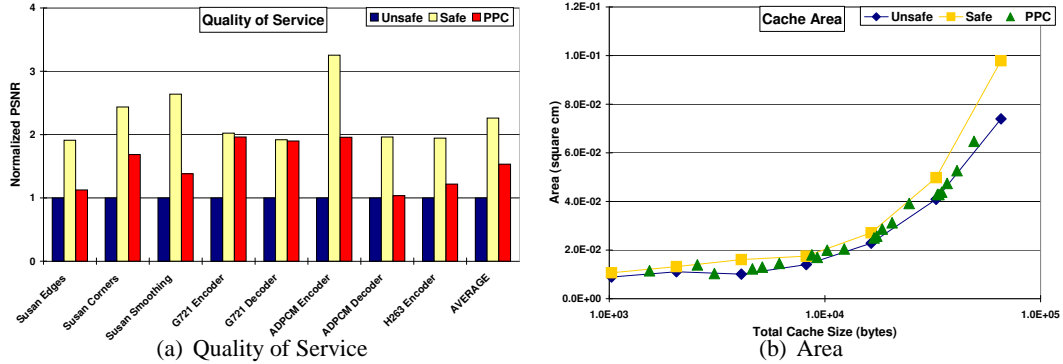


Figure 3.11: Evaluation of Quality and Area

Figure 3.11(a) shows that PPC configuration results in better QoS than the Unsafe configuration and worse QoS than the Safe configuration. Figure 3.11(a) plots the QoS of the three cache configurations normalized to the QoS of the Unsafe cache configuration. The plots show that as compared to the Safe cache configuration, our PPC configuration incurs a QoS penalty of 32%, while as compared to the Unsafe cache configuration our PPC configuration improves the QoS by 54% on an average. Note that QoS values are exaggerated due to accelerated SER in order to observe the failure rates in a reasonable amount of simulation time. Table 3.1 presents the video quality in PSNR using a benchmark *H263 encoder* for the three configurations according to SER, and clearly shows that the quality degradation of PPC configurations is negligible (less than 5% other than  $SER = 10^{-9}$ ) compared to that of Safe configuration.

Figure 3.11(b) compares the area among the Unsafe, Safe, and PPC cache configurations along the total cache sizes. The plot shows that the area overhead for the PPC cache configuration

Table 3.1: Video Quality in PSNR (*dB*) according to SER

SER	Unsafe	Safe	PPC
$10^{-9}$	16.39	31.79	19.67
$10^{-10}$	26.22	33.35	31.89
$10^{-11}$	32.40	33.46	33.40
$10^{-12}$	33.39	33.46	33.45

is smaller than that for the Safe cache configuration since the PPC configuration just implements the SEC-DED algorithm for the only small size of mini cache, which remains the same size for the coding and decoding blocks but reduces the storage size for the control bits. In one specific configuration as 32 KB data cache for the Unsafe, Safe and PPC, which has extra 2 KB protected cache, the area overhead for the Safe cache configuration compared to the Unsafe cache configuration shows about 22% but the PPC cache configuration can be implemented with just 7% area overhead, which means that we can reduce around 12% area when we build the PPC configuration cache instead of the Safe one.

To summarize, our results demonstrate that as compared to the traditional Unsafe cache configuration, our proposed PPC cache configuration can reduce failure rates by  $47\times$ , while incurring only 1% runtime, 3% energy, and 7% area overhead with improved QoS. As compared to the previously proposed Safe cache configuration, our proposed PPC cache configuration can achieve almost the same failure rates while improving both the runtime by 16%, energy by 8%, and area by 12% at the cost of QoS. Thus, PPC architectures allow designers to explore configurations with minimal failure rate by trading-off QoS at minimal power, performance, and area overheads.

### 3.5.2.2 Effectiveness of PPC for General Applications

We perform two kinds of experiments to demonstrate the effectiveness of DPExplore for general applications. In the first set of experiments, we find the page partition with the least vulnerability without any performance loss. Figure 3.12(a) and Figure 3.12(b) plot the vulnerability ratio and the memory subsystem energy ratio, respectively, of the least vulnerability page partition obtained by DPExplore. *Vulnerability Ratio* indicates the ratio of the vulnerability of the *base case* to the vulnerability discovered by DPExplore. Similarly, *Runtime Ratio* and *Energy Ratio* of the least vulnerability page partition obtained by DPExplore are presented in Figure 3.12(b). Thus, each ratio greater than 1 implies the reduction of each metric. In case of no performance penalty, our heuristic algorithm can discover partitions with on average more than 1.2 times reduction in vulnerability, i.e., 1.2 in vulnerability ratio, and only about 3% energy overhead, i.e., 0.97 in energy ratio, over all benchmarks.

In the second experiment, we allow 5% performance degradation. Figure 3.12(c) plots the vulnerability reduction and Figure 3.12(d) plots the increase in energy consumption of the

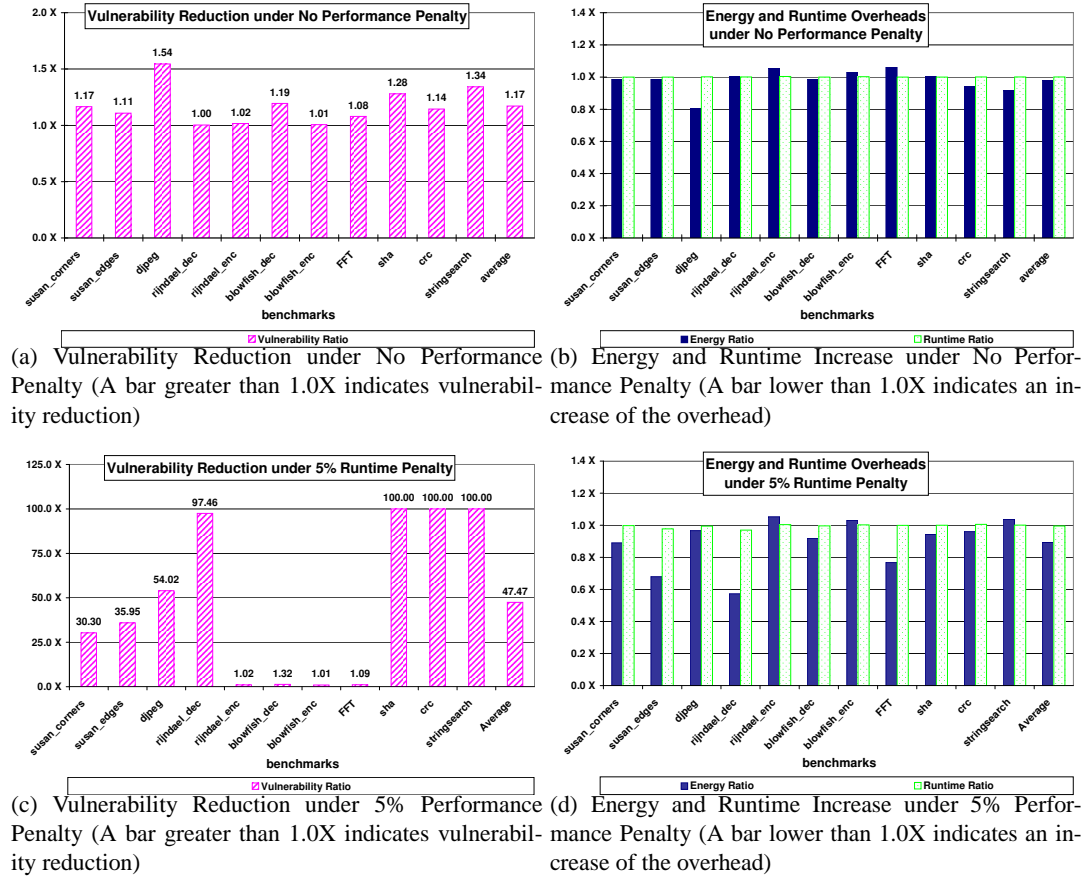


Figure 3.12: Evaluation under *No Performance Penalty* and *5% Performance Penalty*: DPEXplore can significantly reduce the vulnerability at minimal runtime and power overheads

memory subsystem and the increase in runtime of the least vulnerability page partition obtained by DPEXplore. We observe  $47\times$  reduction in vulnerability on average, along with only 0.5% degradation in runtime, and 15% increase in the total energy consumption of the memory subsystem. Compared to the case when all data are mapped to the protected 4 KB cache, i.e., the completely protected cache, the runtime and the energy consumption of the page partition with DPEXplore are improved by 36% and 9%, respectively. Thus, even very small runtime degradation allows DPEXplore to find page mappings that can significantly reduce the vulnerability for general applications.

### 3.6 Summary

Due to incessant technology scaling, soft errors are becoming a critical design concern for system reliability. Especially, soft errors in caches are the most important due to large area and low voltage beyond sub-micron technology. Previously, researchers have investigated the cost-efficient redundancy technique at the hardware or component level while those techniques still incur the overheads in terms of power, performance, and area.

Based on the observation that not all data are equally important in terms of the failure rate, we have studied a cross-layer approach by exploiting the feature of applications or data at the application layer to combat the impact of soft errors at the hardware layer. For example, soft errors on the multimedia data itself do not cause failures (i.e., multimedia data is failure non-critical) while soft errors on other control data such as conditional or loop variables may result in system failures (i.e., control data is failure critical). On the other hand, page partitioning techniques for general applications have been proposed by exploiting the vulnerability of data and codes to separate data and codes for unequal protection.

We proposed a novel cross-layer approach, in which our compiler partitions pages for applications, and PPC architecture provides unequal protection at the hardware layer by exploiting the application features such as the error-tolerance and vulnerability as shown in Figure 3.13. This hardware-software joint solution is very effective to reduce the failure rate or the vulnerability due to soft errors with minimal power and performance overheads in resource-constrained mobile embedded systems.

Our experimental results showed that our technique reduces runtime by 16% and energy consumption by 8% and maintains the same or the even better failure rate in comparison with ECC-protected cache at the cost of QoS degradation for representative multimedia benchmarks. And proposed page partitioning algorithm can effectively and efficiently explore to find page mappings that result in 47 times reduction in vulnerability, i.e., in failure rate, at only 0.5% performance and 15% energy penalty on average in general application benchmarks.

The main contribution of our approach is in extending cross-layer design methodology to combating hardware defects such as soft errors with minimal constraints. This cooperative, cross-layer design methodology can be further expanded to mitigate several types of temporary

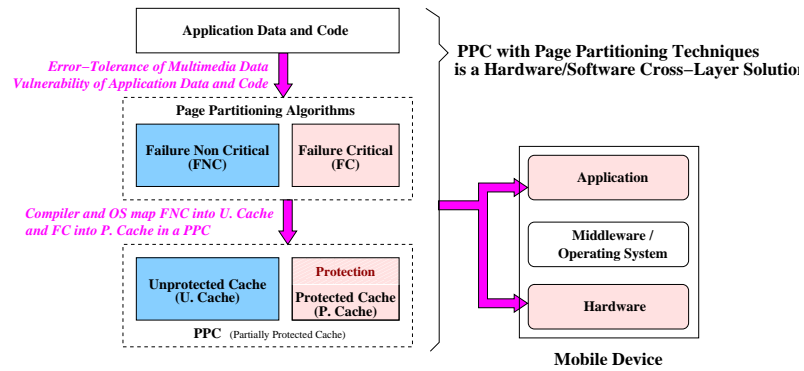


Figure 3.13: Cross-Layer Protection – PPC and Page Partitioning Algorithms provide cost-efficient unequal protection for resource-constrained embedded systems

faults at the hardware layer, e.g., logic components, scratch pad memories, communication architecture, etc, by exploiting the existing features at the application layer.



## Chapter 4

# Error Aware Video Encoding: Enabling Reliability at the Application Layer

### 4.1 Motivation

Due to the rapid deployment of wireless communications, video applications on mobile embedded systems such as video telephony and video streaming have grown dramatically. A major challenge in mobile video applications is how to efficiently allocate the limited energy resource in order to deliver the best video quality or conversely, how to extend the lifetime of video delivery within the limited energy resource without degrading acceptable video quality. A significant amount of power in mobile embedded systems is consumed by video processing and transmission. Also, error resilient video encodings demand extra energy consumption in general to combat the transmission errors in wireless video communications. Thus, it is challenging and essential for system designers to explore the possible tradeoff space and to increase the energy saving while ensuring the quality satisfaction even under dynamic network status. In this chapter, we introduce the notion of *active error exploitation* to effectively extend the tradeoff space between energy consumption and video quality, and present an adaptive error-aware video encoding to maximize the energy saving with minimal quality degradation.

Tradeoffs between energy consumption and QoS (Quality of Service) for mobile video communication have been investigated earlier [24, 37, 82, 83, 114, 131]. It is interesting to observe that the delivered video data is *inherently error-tolerant*: spatial and temporal correlations between consecutive video frames are used to increase the compression efficiency, and result in errors at the reconstructed video data, and also a high quantization scale causes video data losses. These

naturally induced errors and losses from the encoding algorithms degrade the video quality of service, but they may not be perceived by the human eye. In this context, we pay attention to the inherent error-tolerance of video data to increase the energy reduction for resource-constrained embedded systems. For instance, relaxing the acceptable quality of the delivered video reduces the overhead in compression for the exhaustive searching algorithm by exploring a partial area rather than the entire region. Further, we exploit errors actively for the purpose of energy reduction. One way of active error exploitation is to intentionally drop frames before the encoding process. By dropping frames (a process similar to sampling), we eliminate the entire video encoding process for these frames and thereby reduce energy consumption while sacrificing some loss in the QoS of the delivered video stream. Note that the effects of dropping frames on video quality are partially canceled with the nature of error-tolerance in video data.

To cope with transmission errors such as packet losses due to the congested routers and faded access points in wireless communication, error-resilient video encoding techniques [16, 54, 120, 123, 134] have been investigated to reduce the effects of transmission errors on QoS. Most existing error resilient techniques judiciously adapt their resilience levels considering the network status such as packet loss rate. Our approach combines these error-resilient techniques with intentional frame dropping, resulting in several pros and cons. First, we can improve the video quality to the level that error-resilient video encoding techniques achieve by considering these frame drops as packet losses occurring in the network. Second, we can increase the error margins that video encoders potentially exploit for maximal energy reduction, i.e., we can drop more frames. On the other hand, the error-resilience increases the compressed video data in general, and in turn raises the energy consumption for data transmission. As we show in this chapter, this active error-exploitation approach with error resilient techniques significantly enlarges the tradeoff space among energy consumption for compression, energy consumption for transmission, and QoS in mobile video applications. Furthermore, our error-aware video encoding scheme extends the applicability of error resilient schemes, even when the network is error-free. Note that previously proposed error-resilient video encodings were designed to work as a normal video encoding in case of error-free network, i.e., compressing video data as efficient as possible rather than as error resilient, which incurs overheads due to highly complex encoding algorithms.

In this chapter, we discuss a new knob, *Error Injection Rate (EIR)* that controls the

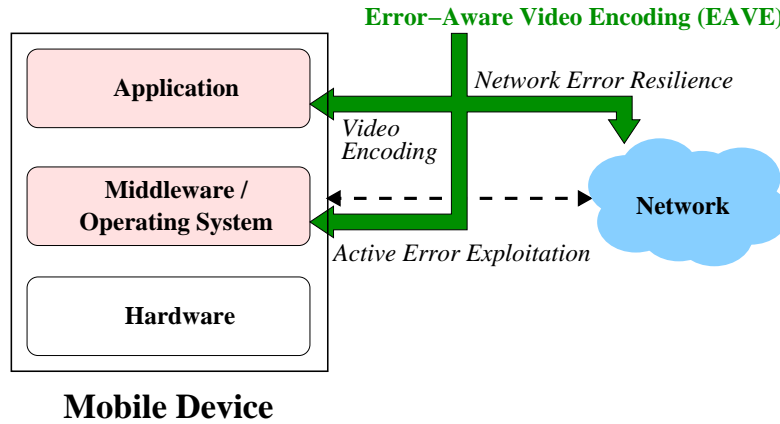


Figure 4.1: Overview of our proposal – EAVE (Error-Aware Video Encoding)

amount of data to be dropped intentionally. This EIR knob can be used to explore the tradeoff space between the energy consumption and video quality, unlike in previous approaches. Specifically, we present an error-aware video encoding technique with EIR based on an existing error-resilient video encoding, PBPAIR (Probability-Based Power-Aware Intra-Refresh) [54]. Our new approach, called Error-Aware PBPAIR or *EA-PBPAIR*, is composed of two units: error-injection unit and error-canceling unit. The error-injection unit drops frames intentionally according to EIR to save the energy consumption. And the error-canceling unit applies PBPAIR to encode video data resilient against intentional frame drops in an energy-efficient manner. Active error exploitation can reduce the overheads for transmission and even the decoding, and results in the energy savings of all components in an encoding-decoding path in distributed mobile embedded systems. However, very aggressive error injection in EA-PBPAIR can degrade the video quality significantly, and there is a need to monitor the delivered video quality in distributed systems and to adjust the error injection rate to ensure satisfactory quality. Thus, we also present *adaptive EA-PBPAIR*, which adapts the error injection rate based on the quality feedback from the decoding side while minimizing the energy consumption.

Figure 4.1 shows the outline of our EAVE (Error-Aware Video Encoding) technique, which develops cross-layer communication between the application and the middleware. The error injection unit (implemented at the middleware layer) monitors the network packet loss rate and the video quality, controls the error injection rate, and translates the sum of packet loss rate and error injection rate for the error rate parameter to the error resilient video encoder at the

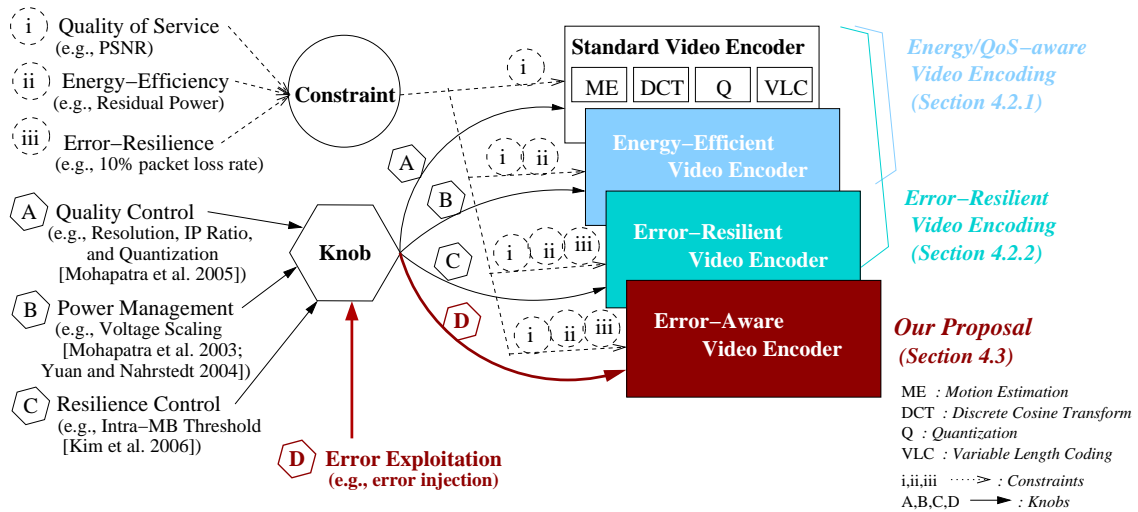


Figure 4.2: Constraints and knobs considered by previous approaches and our proposal

application layer. The error canceling unit is an error-resilient video encoding (e.g., PBPAIR), which is implemented at the application layer. As shown in Figure 4.1, we maximize the energy efficiency of previously proposed error-resilient video encodings by actively exploiting errors, i.e., intentionally dropping frames. Thus, the error-resilience of video encodings at the applications against network errors such as packet losses in the network has been expanded to recover the intentionally induced frame drops at the middleware layer with perspective of the QoS. EAVE or Error-Aware Video Encoding is a cross-layer technique that significantly extends the tradeoff space between energy consumption and video quality for resource-constrained multimedia embedded systems.

## 4.2 Background

### 4.2.1 Energy/QoS-aware Video Encoding

With the increasing popularity of video applications on battery-operated mobile handhelds, energy-efficiency is an essential feature that mobile video applications consider along with QoS. A standard video encoder in Figure 4.2 presents the basic flow of video compression algorithms consisting of ME (Motion Estimation), DCT (Discrete Cosine Transform), Q (Quantization), and VLC (Variable Length Coding). Firstly, the video image is separated into a certain size of data blocks (e.g.,  $8 \times 8$  macro blocks or MB), and each data block is processed through a motion

estimation (ME) algorithm, which exploits the spatial-temporal correlations between video data. After ME, each data block is transformed by a discrete cosine transform (DCT) into its frequency domain equivalent. Then each frequency component is quantized (divided by a quantization scale value) to reduce the amount of data to be transmitted (Q). Finally, these quantized data is encoded using a variable length coding technique (VLC). At each compression step, several parameters need to be selected and each parameter affects the power and QoS. For example, full search and diamond search [116] are two candidates for ME, and they have tradeoffs between energy consumption for computation (diamond search is good since it searches for smaller area than full search), energy consumption for communication (full search is good since it can potentially find the reference data block with smaller difference than diamond search) and QoS (full search is good since it can deliver less difference potentially). Mohapatra et al. [82] explored the effects of video encoding parameters such as quantization scale, IP-ratio, and motion estimation algorithms on energy consumption and QoS.

Energy and QoS aware adaptations have been studied for video applications on mobile handhelds in a cross-layer manner [83, 131]. Mohapatra et al. [83] proposed an integrated power management technique, which identifies interactive parameters among different system levels and tunes them to reduce the power consumption by middleware adaptations aware of system configurations. Similarly, Yuan et al. [131] proposed a global cross-layer adaptation approach, which coordinates CPU, operating system, and application to increase the energy efficiency. Yuan et al. also proposed a practical voltage scaling to minimize the whole system energy of mobile devices while meeting the time constraints of multimedia applications. Eisenberg et al. [24] considered the transmission power along with the video quality at the decoder. To limit the amount of distortion in the delivered video with minimal transmission energy, they exploited the knowledge of the concealment method at the decoder and the relationship between transmission power and the packet loss probability.

Previous efforts have mostly studied the tradeoff between energy consumption and QoS, but they did not take into account error resilience against unreliable transmission and they did not consider active error exploitation.

## 4.2.2 Error-Resilient Video Encoding

Video compression standards such as H.263 [45] and MPEG [86] increase the compression efficiency by exploiting the spatial and temporal correlations among consecutive frames with minimal quality loss. However, these compressed video data can be lost and eventually become error-inclusive at the decoding side through the unreliable channels due to congested routers, link failures, faded access points, etc. in wireless network. Thus, the effects of packet losses are propagated to the following frames due to the nature of spatial and temporal dependency in encoding techniques. To reduce these negative impacts on QoS, several techniques have been proposed and roughly classified into two groups, error-resilient techniques and error-concealment schemes [16]. Typically, error-concealment techniques [27, 121] are implemented at the decoder by recovering the lost data, and error-resilient techniques [16, 54, 120, 123, 134] are designed at the encoder to increase the robustness against the transmission errors by adding the redundancy.

For the purpose of error-resilient video, one of the most effective methods is to introduce the intra-coded frame (I-frame) periodically since I-frames are decoded independently and protect the propagation of the transmission errors in previous frames. We call this video encoding technique as GOP-K (Group-Of-Picture), where K indicates the number of predictively-coded frames (P-frames) between I-frames. For instance, GOP-15 indicates a video encoding technique where one GOP consists of 1 I-frame and 15 P-frames. Recently, Yang et al. [127] reorganized the regular linear GOP structure to decrease the number of descendant frames using double-binary tree structure and thus errors propagate to only few frames. However, the transmission of I-frames causes the delay and jitter due to relatively large size compared to P-frames, and the loss of I-frames is more sensitive on QoS than P-frames [16, 54].

To mitigate both the propagation of the transmission errors and the overheads of large I-frames, intra-MB refresh approaches have been proposed [16, 54, 123]. Mainly intra refresh techniques distribute intra-MBs among frames, and they not only remove the overheads of I-frames but also improve the error-resilience. Worrall et al. [123] introduced the Adaptive Intra Refreshing (AIR), which updates the more important area of MBs more frequently. Cheng et al. [16] allocated intra-MBs on a column-by-column basis in a progressive way considering the residual error propagation, Progressive GOP (PGOP). While most intra-MB refresh techniques

have been focused on alleviating the effects of the transmission errors on the video quality, Kim et al. [54] proposed an energy-efficient and error-resilient video encoding technique named PBPAIR, and presented tradeoffs among error resilience, encoding efficiency, and energy consumption for mobile handheld devices. Note that PBPAIR is not energy efficient in case of low packet loss rates since PBPAIR (like other intra refresh video encoding techniques) is designed to compress the video data as efficiently as standard video encoding.

Most approaches above have focused on *passive error exploitation*, which means that errors are used for relaxing the constraint considering the feature of applications. On the contrary, *active (or aggressive) error exploitation* maximizes the feature of applications even by injecting errors intentionally, which has not been applied previously to video encoding approaches.

#### **4.2.3 Error-Aware Video Encoding: Our Proposal**

While video encoding techniques did not consider error exploitation actively, system designers have recently taken into account errors for their purposes. During system design, since error detection and correction schemes demand high overheads, they exploit the features of applications running on the system they design, and relax the error-correction requirements for the purpose of high yield rate and/or low energy consumption.

Kurdahi et al. [61] proposed an error-aware design scheme for memory subsystems. They observed that strict 100% correctness is not required in some applications such as imaging, video, and wireless communications. They scaled down the voltage level aggressively to the point where the features of those applications can tolerate and let memory system expose errors intentionally; then they achieve significant power savings due to the exponential relation between the supply voltage and the dynamic power dissipation.

In computer networks, Harris et al. exploited packet loss to increase energy-efficiency by discarding the subsequent packets, which compose a larger frame with the lost packet at the application layer (e.g., multimedia data) than a packet at the MAC (Media Access Control) layer [37]. Previously, the frame-induced packet discarding mechanisms were applied to avoid congestion collapse [100], but even in the absence of congestion, they [37] aggressively used the framing-aware link layer mechanisms to reduce the energy consumption, which may be wasted by blindly processing each packet at the MAC layer from the transmission of unusable data at the end.

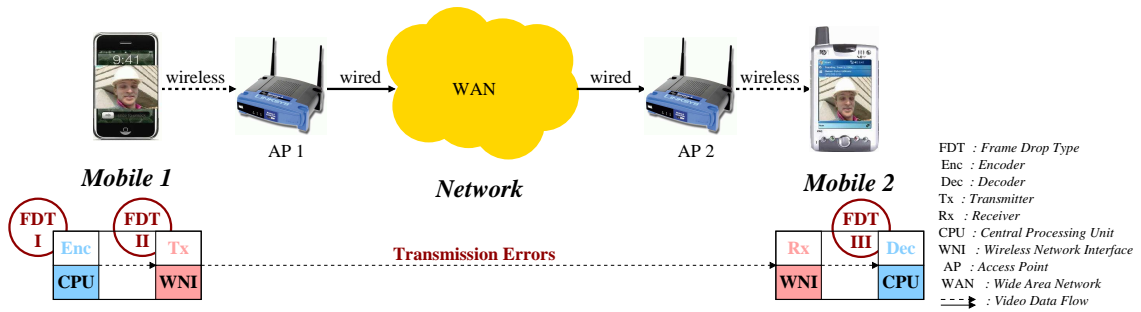


Figure 4.3: System Model (Mobile Video Conferencing) and Frame Drop Types I/II/III for Active Error Exploitation

While these approaches accept errors to their system design or network design, our approach *aggressively exploits* the error tolerance of video data by introducing errors intentionally, and controls the error injection adaptively based on the feedback for the purpose of energy reduction with minimal quality loss for mobile video applications. Therefore, our error-aware video encoding uses errors actively to achieve maximal energy gain while ensuring the QoS and resilience, and further opens opportunities to expand the tradeoff spaces.

### 4.3 Error Aware Video Encoding

#### 4.3.1 System Model

Figure 4.3 shows our system model for mobile video conferencing applications. This mobile video conferencing system consists of two mobile devices (*Mobile 1* and *Mobile 2*) and the network environment (*Network*) between them. The Network consists of WAN (Wide Area Network) and two wireless access points, AP 1 and AP 2, each of which provides the wireless communication channel for each mobile device. To capture the energy consumption for computing and communication, each mobile device is modeled as a mobile embedded system composed of a CPU (Central Processing Unit) and WNI (Wireless Network Interface), where video data is encoded (or decoded) and transmitted (or received). For simplicity, we consider one path from an encoder to a decoder during mobile video conferencing. We analyze the quality of the delivered video at the decoding end, and study each category of energy consumption such as the energy consumption for encoding (Enc EC), for transmission (Tx EC), for receiving (Rx EC), and for decoding (Dec EC). Note that error resilient video encodings have been used to combat transmission



errors such as packet losses induced from the network.

### 4.3.2 Fundamentals of Active Error Exploitation

Recall that we *exploit errors actively*. In our study, active exploitation of errors means intentional frame dropping during video encoding to achieve energy reduction in mobile embedded systems. For the purpose of energy reduction, video frames can be dropped by any of the components in Figure 4.3. For instance, the Decoder can drop the delivered video data to increase the energy reduction for the decoding (*Frame Drop Type III* as in Figure 4.3). Another possible scenario is that the Transmitter can drop video data to save the communication energy, and error resilient techniques take care of the dropped data in advance (*Frame Drop Type II*). Further, the Encoder can drop frames intentionally before the encoding process, and encode the rest of frames robust against the dropped frames, which are considered as lost packets in network (*Frame Drop Type I*).

Note that dropping frames at the Encoder is the most effective in terms of energy reduction since it affects the energy consumption across all the following components in an encoding-decoding path as drawn in Figure 4.3, and the energy consumption for encoding (Enc EC) is relatively high compared to those for the other components in our system model. Therefore, in this particular work, we only consider *Frame Drop Type I* (i.e., intentional frame drop at the Encoder) for active error-exploitation approach. *Type II* and *III* remain as our future work.

Note also that we manage the video quality of service from intentional errors by utilizing the features of error-resilient techniques. Error-resilient video encoding techniques in general incur overheads in terms of power consumption for extra processing and an increase in transmitted data size for the redundancy. Fortunately, we can use a video encoding technique, PBPAIR [54], which is not only error-resilient but also energy-efficient. Furthermore, by dropping frames we can reduce the transmitted data size compared to the original error-resilient video encoders.

Our error-aware video encoder is composed of two units, *error-injection unit* and *error-canceling unit*, as shown in Figure 4.4. The *error-injection unit* controls errors for the purpose of energy reduction, and the *error-canceling unit* reduces the effects of the injected errors using an energy-efficient and error-resilient video encoder. The *Error Controller* acts as an *error-injection unit*, taking into account the constraint (e.g., required video quality) as well as the feedback from

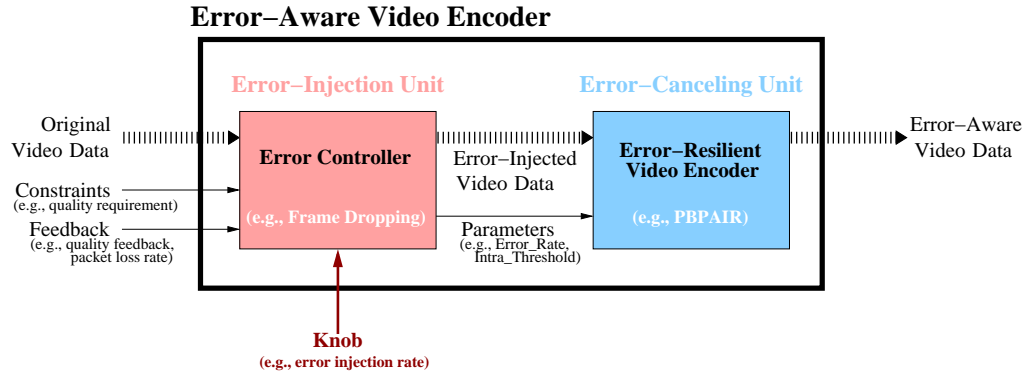


Figure 4.4: Error-Aware Video Encoder is composed of Error-Injection Unit and Error-Canceling Unit with a Knob (Error Injection Rate)

the decoding side (e.g., decoded video quality) and from the network (e.g., packet loss rate). Furthermore, it intentionally injects the amount of errors according to a new knob – error injection rate (EIR), and generates the error-injected video data as illustrated in Figure 4.4. The *Error Controller* also preprocesses parameters for the following video encoder in *error-canceling unit*. Finally, the *Error-Resilient Video Encoder* acts as an *error-canceling unit*, and generates the error-aware video data by encoding the error-injected video data with parameters in preparation for downstream network packet losses as well as intentionally injected errors.

### 4.3.3 EA-PBPAIR: An Error-Aware Video Encoder

We now present *EA-PBPAIR* (Error-Aware PBPAIR), an approach that injects errors intentionally by “Dropping Frames” as an *error-injection unit*, and encodes video resiliently with “PBPAIR” as an *error-canceling unit* as shown in Figure 4.4 and detailed in Figure 4.5.

Dropping frames is one way of injecting errors intentionally. In this study, we consider a simple frame dropping approach, PFD (Periodic Frame Dropping). PFD periodically drops frames according to EIR. For instance, PFD with 10% of EIR drops every 10th frame. PFD evenly distributes the effects of frame dropping on QoS over a video stream. Note that error-exploiting frame dropping does not need to consider the quality since the quality will be deliberately maintained by the nature of error-resilient PBPAIR. Thus, error-exploiting frame dropping in EA-PBPAIR drops any frames within the guaranteed error rate that the original PBPAIR can manage. We show that this simple strategy is effective in demonstrating our error-exploiting approach. Intelligent frame

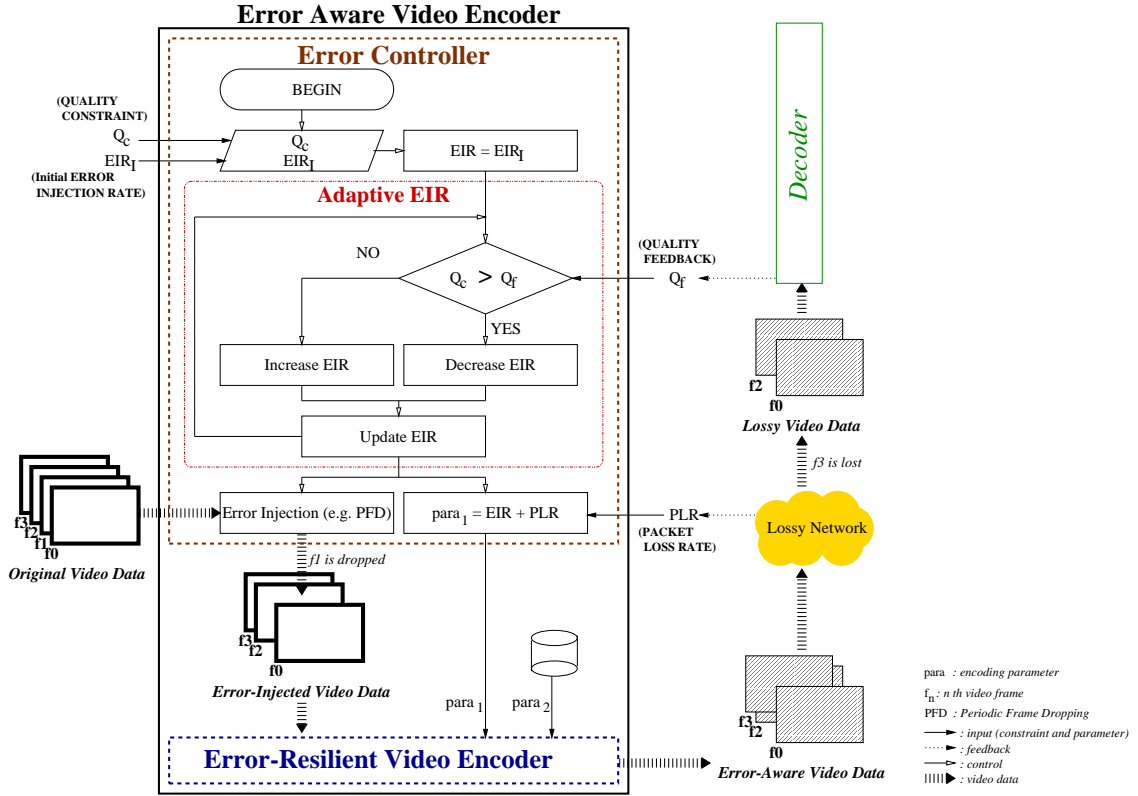


Figure 4.5: Flow of Error Controller and Adaptive EIR in EA-PBPAIR for Mobile Video Applications

dropping strategies can be used as well: these are topics for future research.

We use PBPAIR as an error-resilient video encoder since Kim et al. [54] have demonstrated its energy efficiency while maintaining video quality and robustness against network packet losses. PBPAIR takes two parameters as shown in Figure 4.5. The first parameter ( $para_1 = Error\_Rate$ ) indicates the current network status such as packet loss rate (PLR), and the second parameter ( $para_2 = Intra\_Threshold$ ) represents the level of error resilience requested. To consider both injected errors and packet losses, EA-PBPAIR calculates the sum of EIR and PLR for  $para_1$  ( $para_1 = EIR + PLR$  as shown in Figure 4.5) while PBPAIR originally takes PLR as  $para_1$ . For instance, the first parameter ( $para_1$ ) in EA-PBPAIR is set to 15% when EIR is 10% while PLR in network is 5%.

Note that the active exploitation of errors is orthogonal to PBPAIR and can be applied to any error-resilient and energy-efficient video encoding technique which adapts algorithmic pa-

rameters according to the network status such as packet loss rates. Section 4.4.2.2 will demonstrate the effectiveness of error aware video encodings combining active error exploitation with different video encodings such as GOP and PGOP. PBPAIR inserts more intra-MBs to increase error-resilience, which may increase the encoded video file size while it saves energy consumption by avoiding the computation-intensive motion estimation. Thus, the large video file for error-resilience can increase the transmission energy for data communication. However, intentional frame dropping can reduce this energy overhead, and the energy consumption overhead of EA-PBPAIR for data transmission is relatively small compared to the huge energy saving for the video encoding.

Our error-aware video encoder saves energy consumption in several ways: i) intentional frame dropping saves energy consumption since EA-PBPAIR skips frame encodings according to EIR. ii) the energy consumption for video encoding is reduced since EA-PBPAIR adaptively introduces the more intra-MBs instead of inter-MBs for error resilience due to the intentional frame drops. iii) intentional frame dropping can reduce the encoded video file size, which propagates the energy saving downstream to the Transmitter, the Receiver, and even the Decoder.

#### 4.3.4 Adaptive EAVE

We note that a higher EIR increases the energy reduction for encoding but may decrease the quality of service, if it is beyond a manageable point for error-resilient encoding. To keep the QoS degradation minimal, our approach is able to constrain the EIR based on the feedback from the decoding side in a horizontal cross-layer manner. Figure 4.5 describes this adaptive EIR feature in the *Error Controller*. The *Error Controller* takes the quality constraint  $Q_c$  and sets the initial error injection rate  $EIR_I$ . Then it receives the feedback information such as  $Q_f$  from the decoding side and  $PLR$  from the network as shown in the feedback loop of Figure 4.5, labeled *QUALITY FEEDBACK* and *PACKET LOSS RATE*. If  $Q_f$  is less than  $Q_c$ , the current EIR is bad in terms of QoS, and so the EIR is decreased, and otherwise it is increased (the flow of “Adaptive EIR” in Figure 4.5). Based on EIR, the error injection module periodically drops frames. Thus, the *Error Controller* forwards the error-injected video data instead of the original video data to the *Error Resilient Video Encoder* as shown in Figure 4.5. And  $para_1$  is delivered to the following unit, the *Error Resilient Video Encoder*, which encodes the error-injected video data robust

against the amount of errors indicated as  $para_1$ , with  $para_2$  selected by PBPAIR methodology. Consequently, the encoded video data is now error-aware, i.e., it is cognizant of injected errors as well as packet losses as illustrated in Figure 4.5. This adaptive video encoder adjusts EIR to meet the given quality constraint while minimizing the energy consumption. So we believe that our adaptive approach can be effectively used to adjust our video encoder under a dynamic network environment in distributed embedded systems for maximal energy reduction while ensuring the given quality. Note that strategies for determining the frequencies of feedback (e.g.,  $Q_f$  and  $PLR$ ) are beyond this work, and we assume that feedback channels are reliable.

## 4.4 Effectiveness of EAVE

### 4.4.1 Experimental Setup

For interactive multimedia applications such as mobile video conferencing in distributed embedded systems, an end-to-end experimental system framework is a necessity since all components in a distributed system work interactively and affect other components in terms of energy consumption and performance. Thus, we evaluated EA-PBPAIR on top of an end-to-end framework as shown in Figure 4.6 consisting of a *System Prototype* [48] and NS2 simulator [94] for mobile embedded system and network simulation. The *System Prototype* emulates a PDA (Personal Digital Assistant) and is detailed in our technical report [48].

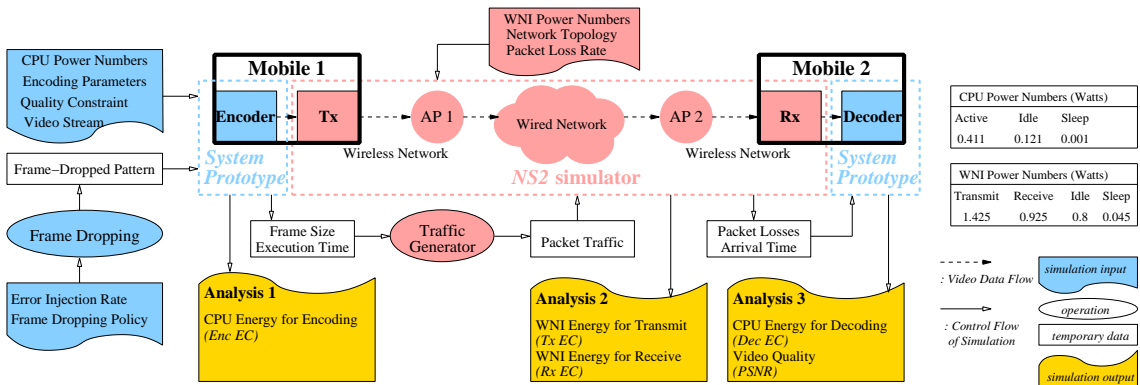


Figure 4.6: Experimental Framework for Mobile Video Conferencing System - *System Prototype* + NS2 Simulator

The left side of Figure 4.6 shows the preprocessing step, where a pattern of frame drop-

ping is generated according to a current EIR. CPU power numbers, video encoder parameters, network status (PLR), and quality constraint are inputs to *System Prototype*, where a video encoder compresses a video stream. *System Prototype* analyzes the first set of results – Analysis 1 – such as the energy consumption for encoding (*Enc EC*), and calculates the encoded size and the encoding completion time of each video frame, which are used for generating the network traffic in the following network simulation. Analysis 1 succinctly shows the CPU energy for encoding at the sender. Next, *NS2* simulates the generated network traffic with a set of configurations including the network topology and WNI power values, and estimates the energy consumption (*Tx* and *Rx EC*) for WNIs – Analysis 2 – at Mobile 1 and Mobile 2 in our system model as shown in Figure 4.3. Thus Analysis 2 captures the end-to-end networking effects, including those of the transmitter and the receiver. Finally at the receiver, the *System Prototype* decodes the transmitted video data based on generated packet losses and frame arrival times from *NS2*, and evaluates the energy consumption for decoding (*Dec EC*) and the video quality estimated in PSNR (Peak Signal to Noise Ratio) in Analysis 3. Thus Analysis 3 captures the CPU energy for decoding at the receiver (Power consumption numbers for CPU [44] and WNI [46] are configured as shown in the tables on the right side of Figure 4.6). By combining Analysis 1, Analysis 2 and Analysis 3, we are able to estimate the entire end-to-end energy savings for our proposed scheme. We now present further details of our experimental framework.

Using *NS2*, we simulate the network consisting of two IEEE 802.11 WLANs (Wireless Local Area Network) and a wired network connecting them as depicted in Figure 4.6. Each WLAN is composed of one access point (AP 1 or AP 2), and one mobile device (Mobile 1 or Mobile 2). We exclude the effects of traffic from other mobile stations in this study since they affect the energy consumption of WNI in our mobile embedded systems. Instead, we limit the data rate of WNI, which constrains the encoded bit rate, and show clearly the effects of the varying data size generated by the Encoder. For the wireless connection, we set the data rate to be 1 Mbps, considered to be an actual data rate [35, 78], and the link layer delay to be 25  $\mu$ s. *NS2* generates packet losses for a given PLR. Each encoded video frame is composed of multiple packets if its size is larger than MTU (Maximum Transfer Unit), which is 1.5 KB in our simulation. A frame is considered lost if any packet of the frame is lost through the network simulation. For each scenario, we simulated more than 100 runs of *NS2* generated pseudo-random packet losses.

Recall that our EA-PBPAIR approach combines PFD (Periodic Frame Dropping) with an existing error-resilient video encoder, PBPAIR. PBPAIR takes two parameters,  $para_1$  and  $para_2$ . We set  $para_1$  (*Error\_Rate*) as the sum of EIR and PLR. For comparison,  $para_2$  (*Intra.Threshold*) is chosen for requested quality with the same compression efficiency as GOP-K (Group-Of-Picture with K) [54]. In this study, GOP-K based on H.263 [45] is defined as a standard video encoder, where K indicates the number of P-frames between I-frames. In GOP-K, we change K for resilience against the transmission errors in network. For example, GOP-3 is selected as a baseline resilient for 10% PLR according to [15, 54]. As test video sequences, *AKIYO*, *FOREMAN*, and *COASTGUARD* in QCIF format ( $176 \times 144$  pixels) are used for our simulation study, and they are typical streams with low activity, medium activity, and high activity, respectively. Note that all video encoders generate a compressed stream at 5 fps (frames per second), which is the maximal frame rate [48] for a typical mobile handheld such as HP iPAQ h5555 [40], and H.263 is designed for low data bandwidth [45, 78] such as 64 kbps (kilobits per second). To constrain the bandwidth, we consider that the bitrate is 64 kbps and frame rate is 5 fps, which keeps the encoders from generating larger than 480 KB for 300 frames of test video sequences.

## 4.4.2 Experimental Results

We present four sets of results. First, we show the energy reduction due to active error exploitation (Section 4.4.2.1). Second, the effectiveness of active error exploitation with different video encoding techniques is demonstrated. Third, we demonstrate the expanded design space allowing better exploration of tradeoff alternatives (Section 4.4.2.3). Finally, in Section 4.4.2.4, we demonstrate the efficacy of our adaptive EA-PBPAIR approach that maintains quality under dynamic network conditions by incorporating feedback on the quality at the receiver end.

### 4.4.2.1 Energy Reduction from Active Error-Exploitation

To show the effectiveness of our proposed technique, the first experiment evaluates EA-PBPAIR with 10% EIR in comparison to GOP-3 considering 10% of PLR in network [15, 54].

Figure 4.7(a) shows the effectiveness of an error-exploiting approach on energy reduction. The plots present the normalized energy consumption and the video quality of EA-PBPAIR to those of GOP-3, and clearly show that EA-PBPAIR is very effective compared to GOP-3 in

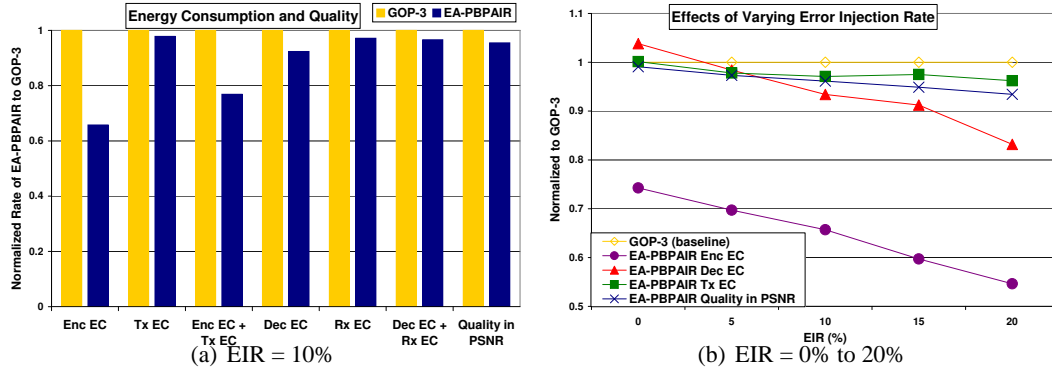


Figure 4.7: Effects of Error Injection Rate on Energy Consumption and Video Quality in EA-PBPAIR compared to GOP-3 (PLR = 10%, FOREMAN 300 frames, Each encoding is constrained with bandwidth)

terms of each category of energy consumption with slight quality degradation. Specifically, EA-PBPAIR consumes 34% less energy than GOP-3 for encoding (Enc EC) since it drops 10% of video frames and compresses more macro-blocks with less expensive intra encodings than predictive encodings. In terms of energy consumption for transmitting video data (Tx EC), EA-PBPAIR sends a similar amount of data within less time than GOP-3, which results in the slight energy reduction. Thus, the energy consumption for the source, including Enc EC and Tx EC, is reduced by 23% with EA-PBPAIR, at the cost of 4% quality degradation in PSNR. Note that 1% quality degradation indicates about 0.31 dB reduction from the PSNR value for GOP-3. At the destination, EA-PBPAIR reduces the energy consumption by 8% for the decoding (Dec EC), which mainly results from dropping 10% frames at the source. Note also that more intra-encoded MB results in more energy consumption for the decoding but 10% frame dropping compensates for this effect. EA-PBPAIR saves the energy consumption for the receiver (Rx EC) by 3% mainly due to the smaller duration for receiving. The energy consumption at the destination (Dec EC + Rx EC) is reduced by 5%. These results are very effective in energy reduction with respect to all energy categories at the cost of slight quality degradation, which is an acceptable tradeoff for power-hungry mobile embedded systems.

We now illustrate how EIR is effective as a knob to tradeoff the quality for energy reduction. To observe the effects of varying EIR on quality and energy consumption, our second experiment compares EA-PBPAIR with GOP-3 by varying EIR from 0% to 20%. Figure 4.7(b) shows the normalized video quality and each energy consumption of EA-PBPAIR to those of



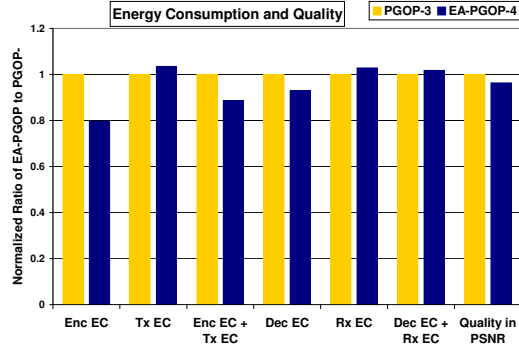


Figure 4.8: Energy Reduction and Quality Degradation of EA-PGOP compared to PGOP (PLR = 10%, EIR = 10%, Error rate is adjusted, FOREMAN 300 frames)

GOP-3. Since we adapt  $para_2$  of PBPAIR to minimize the transmission overhead, the energy consumption for the data transmission (Tx EC) of EA-PBPAIR with varying EIR is close to or less than that of GOP-3. With an increase of EIR, quality is still managed within an insignificant level of quality degradation, and this quality management is mainly because of the error-resilient feature of EA-PBPAIR. With 20% EIR, the loss of quality is 7% in PSNR. In summary, Figure 4.7(b) clearly shows that increasing the EIR significantly saves energy consumption for encoding (Enc EC), with a small reduction in energy for the decoding (Dec EC). Since the portion of intra-MBs for each frame is increasing for error resilience, the energy consumption for the decoding is higher than GOP-3 with low EIR between 0% and 5%. However, with an increase of EIR, the number of frames to be decoded is decreasing and thus the energy consumption decreases. With 20% EIR, we obtain 45% energy reduction for encoding, and 17% reduction for decoding at the cost of 7% quality degradation.

#### 4.4.2.2 Effectiveness of EAVE with different Video Encodings

We compare our EA-PGOP to PGOP in terms of energy consumption and video quality as shown in Figure 4.8. For this experiment, PLR is considered at 10% and PGOP is configured with the number of refresh columns 3 resilient against 10% PLR [16]. EA-PGOP is configured with the number of columns 4 against 10% PLR and 10% EIR. Note that the original PGOP suggests 6 refresh columns per frame against 20% PLR but it substantially increases the portion of intra-MB and results in high energy consumption for the communication. Indeed, our preliminary experiments show that it incurs more than 20% energy overhead for both the transmission and

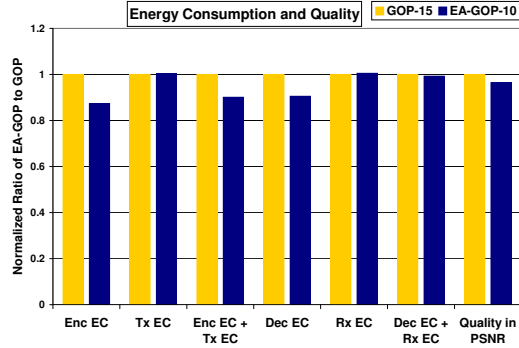


Figure 4.9: Energy Reduction and Quality Degradation of EA-GOP compared to GOP (PLR = 0%, EIR = 20%, Error rate is adjusted, FOREMAN 300 frames)

the receiving. To minimize the energy overhead for the communication, we adjust the error rate for error-resilient video encoding and set the number of refresh columns as 4 rather than 6. This adjustment is able to increase the amount of errors we injected intentionally to save the energy consumption, especially for the communication energy. Figure 4.8 shows that our EA-PGOP saves the energy consumption for the encoding by 20% and for the decoding by 7% while it incurs the energy consumption overhead for the transmission and for the receiving by about 3%, as compared to PGOP. Thus, the source energy consumption is reduced by 11% at the cost of the quality degradation by 4%, while the destination energy consumption is increased by only 2%. This set of experiments demonstrates the effectiveness of active error exploitation in EA-PGOP in terms of the energy consumption for the encoding at the slight loss of video quality.

Next we evaluate the effectiveness of active error exploitation at a standard video encoding, GOP-K. We consider 0% PLR in the network and IP-ratio of GOP-K is set to 15 ( $K = 15$ ). EA-GOP sets 20% EIR and it is configured to generate the similar amount of video data to GOP-15 by adjusting error rate (about -15%). Thus, EA-GOP-10 is selected and the 20% amount of video frames are dropped with PFD. Figure 4.9 shows that EA-GOP-10 saves the energy consumption for the source by about 10% and the energy consumption for the destination by about 1% at the cost of 4% video quality degradation. So this set of experiments demonstrates the effectiveness of active error exploitation with a standard video encoding such as GOP-K in terms of the energy consumption at the slight degradation of the QoS.

In summary, these experiments clearly demonstrate the effectiveness of our error-aware

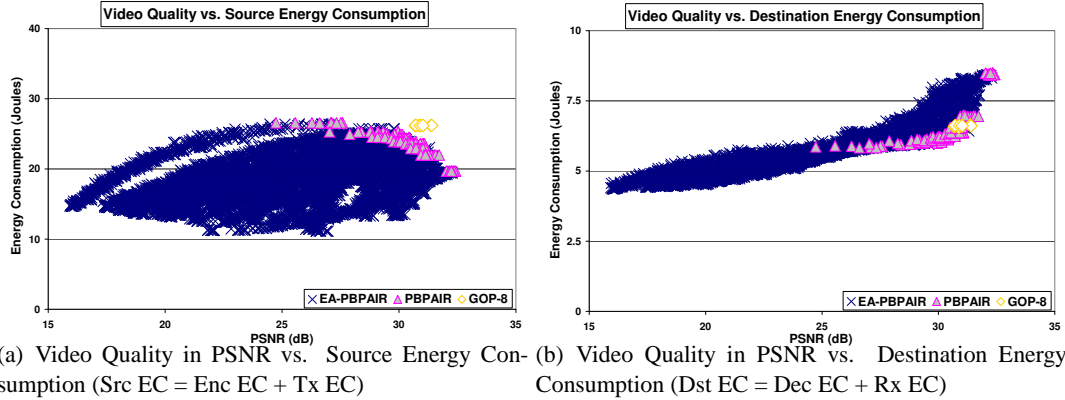


Figure 4.10: Extended Tradeoff Space between Video Quality and Energy Consumption by EA-PBPAIR in comparison to GOP-8 and PBPAIR (EIR = 0% to 50%, PLR = 5%, FOREMAN 300 frames, Each encoding is constrained with bandwidth)

video encoding, EAVE, in terms of the energy consumption by trading off the video quality in several video encoding techniques.

#### 4.4.2.3 Extended Energy/QoS tradeoff

To show the effectiveness of EA-PBPAIR on extending the tradeoff space between energy consumption and QoS, we performed simulations by varying the EIR and the error resilience ( $para_2$ ). We increase the EIR at the Encoder from 0% up to 50% in 1% increments and  $para_2$  from 0% to 100% in 10% increments, and observe the effects on the energy consumption and quality.

Figure 4.10(a) plots the energy consumption at the source (Src EC) vs. quality of EA-PBPAIR compared to PBPAIR and GOP-8 for 300 frames of a test bitstream, *FOREMAN*, and clearly shows that design space of EA-PBPAIR is much larger and more effective than those of PBPAIR and GOP-8. As compared to PBPAIR, the tradeoff space of EA-PBPAIR subsumes all spaces for PBPAIR since indeed EA-PBPAIR with 0% of EIR is PBPAIR. As compared to GOP-8, EA-PBPAIR generates a better design space in terms of the energy consumption without losing video quality, and presents even better video quality with less energy consumption. Further, relaxing the quality requirement (such as 10% QoS degradation) compared to GOP-8 increases the energy reduction at the source by up to 49%. Thus, EA-PBPAIR very effectively expands the design space between the source energy consumption and video quality by exploiting intentional errors. Figure 4.10(b) depicts the tradeoff space between the energy consumption at the destination

Table 4.1: Energy Reduction at the Cost of Video Quality

QoS Degradation	(a) Energy Saving at Source			(b) Energy Saving at Destination		
	0%	5%	10%	0%	5%	10%
<i>AKIYO</i>	49%	51%	51%	9%	22%	28%
<i>FOREMAN</i>	37%	48%	49%	3%	10%	11%
<i>COASTGUARD</i>	13%	15%	26%	1%	4%	9%

(Dst EC) and the video quality, and clearly shows that EA-PBPAIR greatly extends the spaces explored by PBPAIR and GOP-8. However, the energy saving at the destination using EA-PBPAIR is less effective than that at the source since the resilience approach encodes more intra-MBs, which decreases the energy saving resulting from the intentional error injection. Even then, EA-PBPAIR can save the energy consumption by 3% without losing QoS compared to GOP-8.

Simulations for different video streams such as *AKIYO* and *COASTGUARD* presented similar results, i.e., the error exploiting video encoding is very effective to extend the tradeoff space between the energy consumption and the video quality (as detailed in our technical report [48]). Table 4.1 summarizes how much energy can be saved with EA-PBPAIR based on these profiled experiments at the cost of quality, and clearly shows that EA-PBPAIR is very effective in terms of energy reduction, especially in terms of the energy reduction at the source. While EA-PBPAIR achieved energy saving by up to 49% at the source without quality degradation for *AKIYO*, a smaller amount of energy reduction (13%) is observed for *COASTGUARD*. A video clip with low activity such as *AKIYO* increases the energy reduction more effectively while minimizing the quality loss. These results are because *AKIYO* has high correlation between frames, and thus it helps EA-PBPAIR drop as many frames as possible within the given quality constraint. On the other hand, in bitstreams with high activity such as *COASTGUARD*, dropped frames can propagate errors dramatically due to high correlation among consecutive frames in *COASTGUARD* compared to *AKIYO* and *FOREMAN*. Thus, dropping frames in bitstreams with high activity requires additional strategies to obtain the further energy reduction while ensuring the quality requirement.

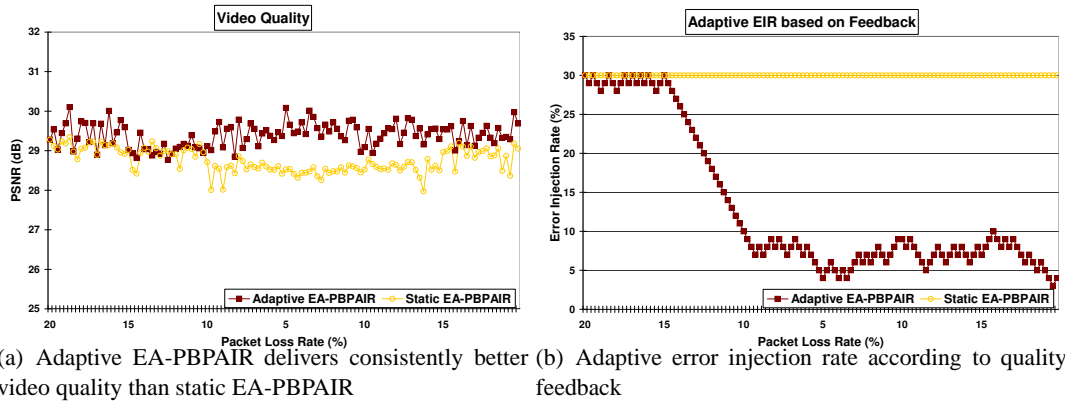
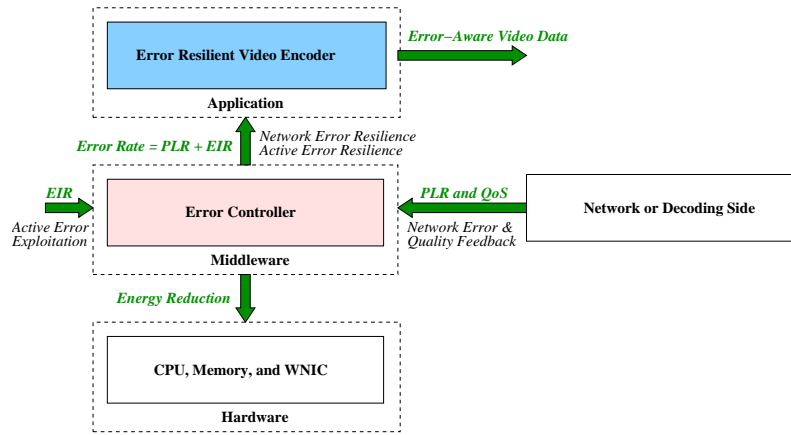


Figure 4.11: Adaptive EA-PBPAIR Robust to Varying PLR under Dynamic Network Status

#### 4.4.2.4 Adaptive EA-PBPAIR: Ensuring Quality under Dynamic Network Status

To show the effectiveness of our adaptive EA-PBPAIR by updating EIR, we model a dynamic network and compare adaptive EA-PBPAIR to static EA-PBPAIR (i.e., EA-PBPAIR with a fixed EIR). For this experiment, PLR begins with 20% and decreases by 5% every 20 runs and after 5% PLR it increases by 5% until it reaches 15%. Each run captures 300 frames of video encoding. The horizontal axis in Figure 4.11 represents this scenario with varying PLR. The quality constraint is set to 29.6 dB in PSNR, which is about 10% quality degradation from GOP-3 without any errors and losses. Static EA-PBPAIR encodes the video data with a fixed EIR = 30% (since 30% EIR degrades the video quality significantly in some dynamic network situations as shown in Figure 4.11(a)) while adaptive EA-PBPAIR starting with 30% EIR and updates it according to the quality feedback. Figure 4.11(a) draws the PSNR values for adaptive EA-PBPAIR in comparison to static EA-PBPAIR, and shows that the delivered quality of adaptive EA-PBPAIR is consistently better than that of static EA-PBPAIR. EA-PBPAIR adapts the EIR according to the feedback with respect to the video quality as shown in Figure 4.11(b). The important observation we can make from Figure 4.11 is that adaptive EA-PBPAIR can adjust EIR dynamically to keep the quality considering the minimal energy consumption. In conclusion, this EIR adaptive technique with EA-PBPAIR adjusts the quality of service based on the feedback for mobile video applications in distributed embedded systems while minimizing the energy consumption.



Error-Aware Video Encoding (EAVE) at Mobile Device

Figure 4.12: Cross-Layer Error-Exploitation – EAVE maximally exploits the energy efficiency and error resilience of previously proposed video encodings by intentionally injecting errors for resource-constrained embedded systems

## 4.5 Summary

Mobile video applications pose significant challenges for battery-constrained embedded systems due to high processing power for compression algorithms and transmission of a large volume of video data. Fortunately, video applications tolerate errors inherently, and we exploit this error tolerance of video data for the purpose of the energy reduction. Active error exploitation – intentional frame dropping together with error-resilient video encoding – can achieve significant energy gains while ensuring satisfactory video quality. We present a new approach where errors can be intentionally injected to balance the dual goals of energy efficiency and satisfactory QoS.

Figure 4.12 shows the outline of our cross-layer error-aware video encoding to tradeoff the video quality for the system energy reduction in resource-limited embedded systems. We have developed a new knob, EIR or Error Injection Rate, and active error exploitation composing Error Controller at the middleware layer and Error Resilient Video Encoding at the application layer. Also in the middleware layer, the feedback mechanism monitors the delivered video quality by communicating the decoding side and enables the Error Controller to adjust the error rate to an error-resilient video encoding at the application layer. Now, an error-resilient video encoding takes into account not only network errors but also active (intentional) errors, and thus is able to provide both the network error resilience and the active error resilience as shown in Figure 4.12. This

active error exploitation eventually reduces the energy consumption at the hardware layer such as CPU and WNI energy savings.

In this chapter, we demonstrated our cross-layer approach, EAVE, in two phases for video conferencing applications running on resource-limited mobile systems. First, we presented EA-PBPAIR that combines an error-resilient video encoder (PBPAIR) with intentional frame dropping to significantly reduce the energy consumption for the entire encoding-decoding path of the video conferencing application. We also presented the effectiveness of EAVE by exploiting errors intentionally with different video encodings such as PGOP and GOP. Our experiments demonstrated that the active error exploitation of EA-PBPAIR allows system designers to consider larger tradeoff spaces than previous approaches: GOP and PBPAIR. Further, we proposed an adaptive EA-PBPAIR by controlling a new knob EIR in order to satisfy the delivered quality based on the feedback under the dynamic network status.

## Chapter 5

# Cooperative Cross-layer Protection

### 5.1 Motivation

As described in earlier chapters, combating soft errors in modern mobile multimedia devices is extremely challenging, owing to the multi-dimensional design requirements. Traditional reliability techniques attempt to provide the entire “fix” at one level, e.g., error correction codes (ECC) at the hardware level, packet retransmission at the network level, and triple modular redundancy (TMR) at the component level, and consequently have extremely high overheads. For example, trying to correct all the errors in hardware itself requires data encoding using an ECC scheme, which incurs very high power and performance overheads. For instance, implementing an ECC-based scheme raises access time by up to 95% [70] and power consumption by up to 22% [95] in the caches. Clearly such high overheads are not acceptable for mobile embedded devices (such as PDAs) as they are extremely sensitive to the power, performance, and cost overheads.

Cross-layer techniques distribute the functionality across different design abstraction layers and exploit the best features of each layer, with the goal of achieving flexible and efficient design solutions. Cross-layer approaches for multimedia have been used in a variety of previous contexts primarily for power and QoS. For instance, GRACE [34, 130] deploys cross-layer methods for maximizing power reduction with the satisfactory QoS; DYNAMO [23, 31, 83, 82] proposes a proxy-based middleware approach for trading off video QoS and power; and xTune [53] performs cross-layer online timing-QoS verification at the proxy server. Unlike these approaches, our focus is to provide a cross-layer strategy for achieving reliability, and trading off reliability for power/QoS in mobile devices. Our goal is to coordinate approaches among abstraction layers to



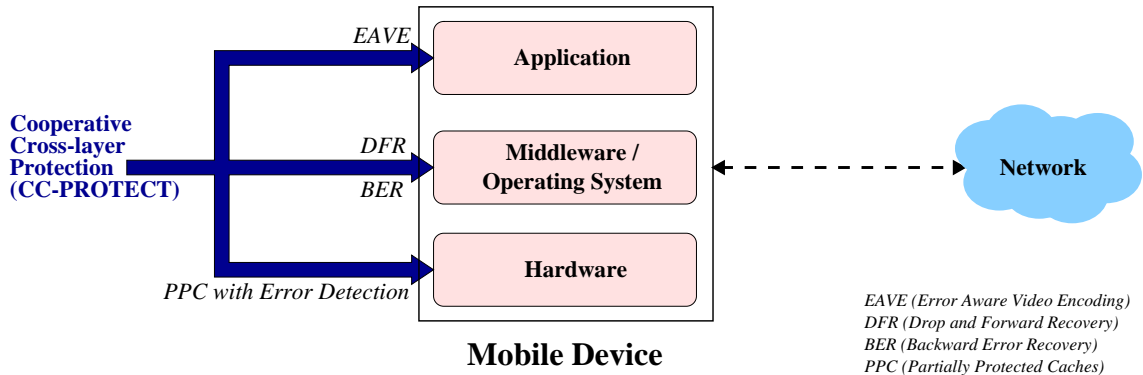


Figure 5.1: Overview of our proposal, CC-PROTECT

find the best cross-layered scheme that achieves the maximal reliability with minimal overheads. Integrating approaches unaware of interactions among them across layers is not efficient since there exist conflicts among them with respect to multiple properties, and also it may cause over-protection or under-protection. For example, protecting all data in memory systems is an overkill in multimedia applications since we may not need to protect multimedia data that do not cause failures in general [65].

In this Chapter, we present CC-PROTECT (Cooperative, Cross-layer Protection); Figure 5.1 shows the outline of our proposal with cross-layer perspectives. We observe that error detection is much cheaper than error correction in hardware. Therefore, we perform only error detection in hardware using error detection codes (EDC) for previously presented PPC (Partially Protected Cache) [65]. For automated recovery of the detected errors, we deploy error recovery solutions in the middleware. Traditionally, on receiving an erroneous frame, the middleware will request re-transmission of the frame. We call this scheme *Backward Error Recovery (BER)*. In contrast, a *Drop and Forward Recovery (DFR)* mechanism drops the erroneous frame, and reconstructs it by using data from adjacent frames [121] (Section 5.3.2). While BER can result in significant power and performance overhead, DFR can result in significant loss in QoS. Therefore, we present and explore hybrid approaches of using DFR and BER to achieve low overheads in power and performance without much loss in QoS (Section 5.4). Furthermore, we exploit an error-aware video encoding technique at the application layer to improve the QoS based on Probability-Based Power Aware Intra Refresh or PBPAIR [54] (Section 5.3.3). Our cross-layer approach is also able to exploit specialized microarchitectural features for reliability (e.g., a Partially Protected Cache

or PPC) in a seamless manner (Section 5.3.1). We show that this cross-layered approach is effective in increasing the reliability of common multimedia streams (and simultaneously improving performance and reducing energy consumption), with minimal quality degradation as compared to a low-level hardware-based error correction scheme (Section 5.5.2).

## 5.2 A Cross-Layer Approach to Support Reliability and QoS

### 5.2.1 System Model and Problem Definition

In the previous section, we argued the increasing need for reliability in mobile applications. It is well understood that mobile multimedia applications such as video streaming and conferencing applications have soft real-time constraints on data delivery. Missing deadlines in video streaming applications results in service delay and packet losses that degrade the video quality. In practice, such degradation (when perceivable) is acceptable to some extent by end-users based on the nature of the application. While we can exploit the soft real-time nature of these applications and their tolerance to slight quality degradation, our ability to do so is already limited in the mobile execution environment that is resource-constrained (limited buffering and battery, error-prone networks, etc).

Techniques have been developed to enable QoS in multimedia applications executing in error-prone networks. Error-resilient video encoding techniques enable adaptive encoding of information based on knowledge of network conditions [16, 54]. Applications may also selectively tag data with their level of importance; in-network mechanisms use the tags to selectively drop information when system or network conditions change. Note that these techniques aim to protect the multimedia content that flows through error-prone networks. We refer to this multimedia content as *external data*, i.e., the payload on which the application is executed. In contrast, *internal data* is defined as data, program code, etc. residing inside the mobile device during the process of execution and representing the programs/data that implement the application functionality, e.g., the video codec and associated data/variables.

The key observation is that while errors in external data (due to packet losses etc.) only cause quality degradation of the multimedia stream, errors in internal data may cause not only QoS degradation but also system failures. In particular, defects induced at the hardware layer, e.g., soft

errors in data caches, manifest themselves differently as compared to network errors on external data. In general, errors on internal data, especially on control data or program variables, can result in system crashes, infinite loops, and memory segmentation faults - leading to application failures.

Hardware error-protection techniques can be designed to protect internal data from hardware failures. Traditional protection techniques such as TMR and ECC [96] implemented at the hardware layer to combat such transient errors incur significant overheads in terms of power, performance, and cost. For example, PPC (Partially Protected Caches) [65] utilizes knowledge of content and device hardware capabilities to selectively place critical data in more reliable hardware (e.g., a protected cache), but it still incurs overheads of power and performance at the protected cache in a PPC.

In this chapter, our goal is to exploit the limited error tolerance of mobile multimedia applications to enhance their reliability to hardware-level "faults" without creating an adverse impact on power and performance profile at the device level or sacrificing application QoS. *We believe that addressing such power, performance, reliability, and QoS tradeoffs in the presence of hardware faults requires a cross-layer approach.* Firstly, we need to develop an understanding of how errors occur at the various layers and understand existing mechanisms that have been developed to avert errors. This will then enable us to determine when "errors" become "failures" and how "failures" manifest themselves at various system layers. We can then design appropriate schemes at different layers to prevent/bypass specific failures and detect/recover from them.

Table 5.1 presents different error models and error control schemes at the application and hardware abstraction layers in a mobile multimedia system. By being aware of error specifics and error control schemes, we expect that systems can be designed in a cross-layered manner for obtaining low-cost reliability while maintaining the QoS. A closer look at Table 5.1 reveals

Table 5.1: Error models and error control schemes at different abstraction layers

Abstraction Layer	Application Layer	Hardware Layer
Error Model	Packet Losses	Soft Errors
Data Perspective	<i>External Data</i>	<i>Internal Data</i>
Impacts	Quality Degradation	Quality Degradation and <b>System Failure</b>
Protection	Error-Resilience, Error-Concealment, etc.	Triple Modular Redundancy, Error Correction Codes, etc.
Error Metric	Packet Loss Rate (%)	<b>Soft Error Rate (FIT<sup>a</sup>)</b>

<sup>a</sup>FIT (Failures In Time): the number of failures in  $10^9$  operation hours

that while errors occur dynamically and in a transient fashion, techniques to combat these errors may be static or dynamic. For instance, the PPC approach uses compiler-assisted techniques to statically tag data; the operating system uses the tags at runtime to stage the data appropriately into a protected cache. Expensive error correcting code (ECC) mechanisms are then employed on the protected data cache to ensure the reliability of information stored in the cache, *irrespective of whether the error rate is high or low*. Dynamic schemes periodically checkpoint memory state and use knowledge of current error levels, captured via the soft error rate (SER) metric, to trigger rollback to the checkpoints. Given the dynamic nature of multimedia data and real-time needs of multimedia applications, this approach as a sole method to deal with soft errors requires very frequent checkpointing and is hence impractical.

### **5.2.2 Related Work**

Existing work already demonstrates the effectiveness of cross-layer methods for mobile multimedia as opposed to schemes isolated at a single abstraction layer [53, 83, 82, 130]. Yuan et al. [129] proposed an energy-efficient real-time scheduler (GRACE-OS) based on statistical distribution of application cycle demands, and presented a practical voltage scaling algorithm [130] to coordinate adaptation of multimedia applications and CPU speeds for mobile multimedia systems. Mohapatra et al. [83] presented an integrated power management technique considering hardware-level power optimization and middleware-level adaptation to minimize the energy consumption while maintaining user experience of video quality in mobile video applications. Recently, Kim et al. [53] proposed a unified framework that allows coordinated interactions among sub-layer optimizers through constraint refinement in a compositional cross-layer manner to tune the system parameters.

Cross-layer methods in the OSI (Open Systems Interconnection) reference model have been widely investigated as a promising optimization tools to efficiently reduce the transmission energy consumption in wireless multimedia communications [7, 117, 118]. Vuran et al. [118] presented a cross-layer methodology to analyze error control schemes with respect to transmission power and end-to-end latency, especially impacts of routing, medium access, and physical levels in wireless sensor networks. Schaar et al. [117] proposed a joint cross-layer approach of application-layer packetization and MAC-layer retransmission strategy, and developed on-the-fly adaptive

algorithms to improve the video quality under the bandwidth and delay constraints for wireless multimedia transmission. Bajic [7] developed cross-layer error control schemes considering joint source rate selection and power management for wireless video multicast.

Our work presented in this chapter is novel in two respects. First, we address a broader notion of reliability than has been explored for error-resilient multimedia applications by specifically focusing on hardware induced defects (soft errors) and their impacts. As illustrated earlier, this issue is a leading concern for embedded architectures of the future. Secondly, we present how to exploit the cross-layer methodology to activate error control schemes at one abstraction layer to combat errors at a different abstraction layer.

### 5.2.3 Cooperative Cross-Layer Approach

We conjecture that a dual pronged approach is needed to effectively address the aforementioned tradeoffs among power, performance, reliability, and QoS. Firstly, *error-awareness* is critical to selectively trigger reliability mechanisms when errors occur - this can be achieved through suitable monitoring mechanisms that determine hardware errors (that can potentially cause failures). Secondly, the monitored errors are used to tailor intelligent compositions of error-protection schemes across layers using *cooperative, cross-layer* schemes. Specifically, we focus on transient hardware errors (soft errors), i.e., they do not *immediately* cause a permanent failure of the system. To create error-awareness, we consider the presence of inexpensive error detection mechanisms for soft error detection - these schemes generate as output the soft error rate (SER), which is translated into an error rate for error control schemes described in Section 4.2.2. Hence, our problem is to develop cross-layer methods that, given dynamic soft error rates, are capable of: (i) minimizing the overheads of power and performance, (ii) satisfying the QoS requirement, and (iii) achieving the same level of fault tolerance as traditional error protection techniques. In particular, we investigate techniques to exploit error-resilient video encoding mechanisms (at the application layer) and selective recovery mechanisms (applied at the middleware layer) to reset potentially harmful data in memory (at the hardware layer).

To illustrate and evaluate our cooperative cross-layer approach, we consider a simplified system consisting of a video encoding application and a data cache as shown in Figure 5.2. Video encoding can be *error-prone* or *error-resilient*; a data cache can be *error-prone* or *error-protected*.

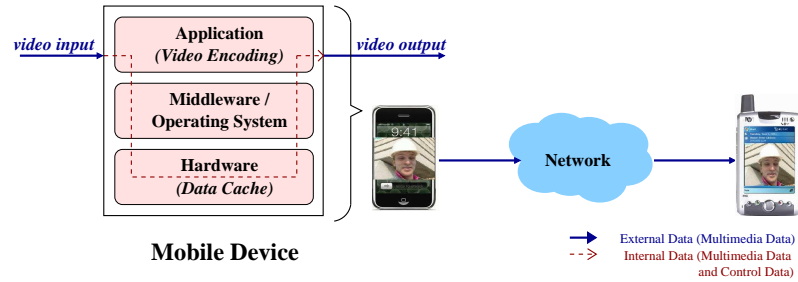


Figure 5.2: System Model - Mobile Video Encoding System

Different compositions vary with respect to overall performance, power, reliability, and QoS. For example, error-prone video encoding executing on an error-prone data cache suffers from high failures due to no protection at data cache against soft errors. Adding error-protected data caches improves the video quality as well as the reliability, however it incurs high overheads in terms of power and performance. Error-resilient video encoding on an error-prone data cache may increase the video quality due to the feature of error resilience, but fail to increase the reliability. An error-resilient video encoding running on an error-protected data cache is possibly of over-protection on the QoS and it incurs high overheads due to expensive protection.

Given the ability to support error-awareness through less expensive error detection codes based schemes, our strategy is to use the information on SER (soft error rate) to

1. bypass potential failures by triggering error recovery mechanisms which reinitialize the erroneous data cache, and
2. reinforce application data using error-resilient encoding mechanisms by translating the SER into the input metric of the encoding algorithm (being considered as the network packet loss rate).

In other words, awareness of micro-level errors (i.e., bit errors) is translated into policies that have macro-level impacts in terms of execution failure, performance, energy consumption, and QoS.

In particular, we explore a Drop and Forward Recovery (DFR) mechanism (shown in Figure 5.4(c)) that drops a current encoding frame and moves forward to the next frame once an error is detected in a mobile video encoding system. The DFR mechanism works effectively with an EDC scheme to improve power and performance significantly while increasing reliability as well. As discussed, EDC is much less expensive than ECC [72] and overheads due to checkpoints

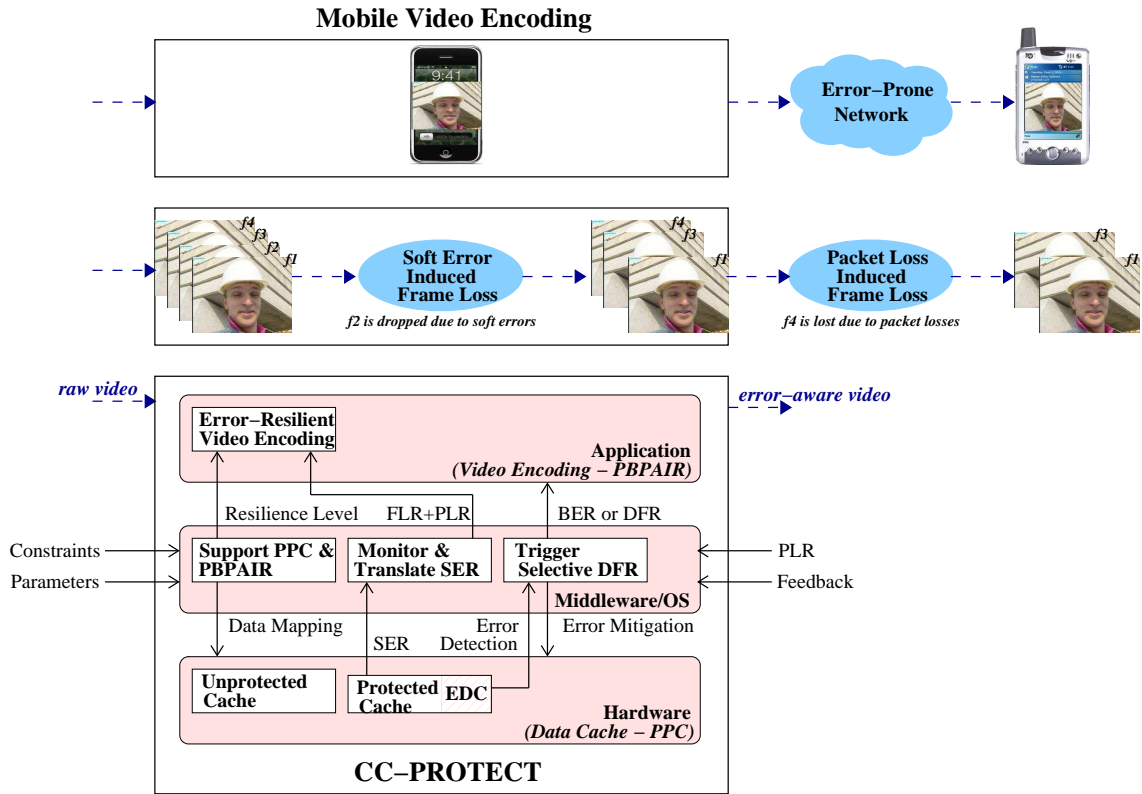


Figure 5.3: CC-PROTECT (Cooperative Cross-layer Protection) – mitigating hardware defects with minimal costs by using error-resilience and a Drop and Forward Recovery (DFR) in a video encoding

are negligible [126] while EDC can be as immune to soft errors as ECC with respect to reliability. Further, dropping an erroneous frame potentially improves performance and energy reduction since it skips expensive processing algorithms for encoding the frame.

However, just using DFR-based mechanisms can result in video quality degradation since erroneous frames are actually dropped. To some extent, these errors can be recovered by wisely injecting error-resilience at the application layer. To enhance QoS, we also explore the selective use between DFR and Backward Error Recovery (BER) mechanism that rolls backward and re-encodes the current frame once an error is detected as shown in Figure 5.4(b).

## 5.3 Cooperative Cross-Layer Protection (CC-PROTECT)

In this Section, we present CC-PROTECT - a middleware driven approach for cooperative composition of cross-layer strategies to support error resilience. Figure 5.3 illustrates our CC-PROTECT scheme which exploits the error-resilience of video encoding along with DFR-based error recovery mechanisms to mitigate the impacts of soft errors at the hardware layer.

We instantiate two specific strategies for error recovery and error-resilience within CC-PROTECT. The specific error metric we use and evaluate in this chapter is soft error rate (SER). First, to mitigate the impact of soft errors on the video quality, we exploit a power-aware error-resilient video encoding technique, PBPAIR [54]. We present a simple, intuitive, and effective translation of SER into frame loss rate (FLR) used in turn by the error-resilient PBPAIR. Next, we exploit our prior work (partially protected caches or PPC [65]) to design a naive DFR mechanism. Using information captured in the middleware, we then extend the naive mechanism to achieve a balance between DFR and BER in Section 5.4.

### 5.3.1 Error Detection at Hardware

A PPC architecture consists of two caches at the same level of memory hierarchy for unequal data protection – we refer to them as the unprotected cache and the protected cache. Typically, hardware-based ECC techniques are applied on the protected cache. In our design, the protected cache is equipped with an EDC that only detects errors and hence improves power and performance as compared to an ECC-equipped cache [72]. Since multimedia data itself does not cause a system failure [65], multimedia data is exposed to soft errors by being mapped into the unprotected cache in a PPC. And the rest of data is mapped into the protected cache in a PPC. Mapping data into two caches in a PPC is managed by the operating system as shown in Figure 5.3. To correct an error, we use a drop and forward recovery at the middleware layer. Thus, the error detection in the protected cache will be reported to the middleware, and mitigation technique will handle it as shown in Figure 5.3. Also, to manage the quality degradation due to frame drops induced by soft errors on control data in the protected cache, soft error rate is informed to the middleware as shown in Figure 5.3. Now CC-PROTECT implements the data protection using a PPC with an EDC scheme at the minimal costs of performance and power at the hardware layer.



### 5.3.2 Drop and Forward Recovery at Middleware

Soft error rates, obtained by error detection techniques at the hardware layer, are communicated to the middleware as shown in Figure 5.3. The middleware then:

1. monitors errors, and maintains execution histories and video quality information,
2. translates SER values to corresponding metrics used by other policies (frame loss rate or FLR in our case),
3. initiates DFR/BER policies (discussed later) to avoid and bypass potential hardware failures, and
4. adaptively fortifies multimedia content when hardware errors occur by triggering error-resilient encoding at the application layer.

Traditional error recovery techniques can be classified into Forward Error Recovery or FER (e.g., ECC) and Backward Error Recovery or BER (e.g., Checkpoints) [96] according to when an error is recovered as shown in Figure 5.4(a) and Figure 5.4(b), respectively. We explore the use of Drop and Forward Recovery or DFR (See Figure 5.4(c)) that combines error detection mechanisms with checkpoints to discontinue processing of the current frame and to initiate processing of the next checkpointed frame.

Our objective is to apply DFR techniques on a cache architecture that uses a PPC (partially protected cache). We first present a naive implementation of DFR in the PPC architecture. Here, checkpoints are taken just before the starting operation where each frame  $K$  is encoded (similar to BER). The only difference is that DFR must save the required values for the next encoding frame (frame  $K + 1$ ) in Figure 5.4(c). Whenever an error is detected on the (control) data in the protected data cache by the EDC mechanism in a PPC, content for the next frame encoding is loaded into the protected data cache with the help of the operating system averting a memory-based system failure. The expectation is that generally a frame drop induced by DFR does not cause a significant quality loss mainly due to the inherent error-tolerance of video data [63]. First, we exploit the error-resilient video encoding technique to recover the possible degradation of the quality due to frame drops. We next discuss extensions to the naive DFR scheme to overcome quality losses when they occur.

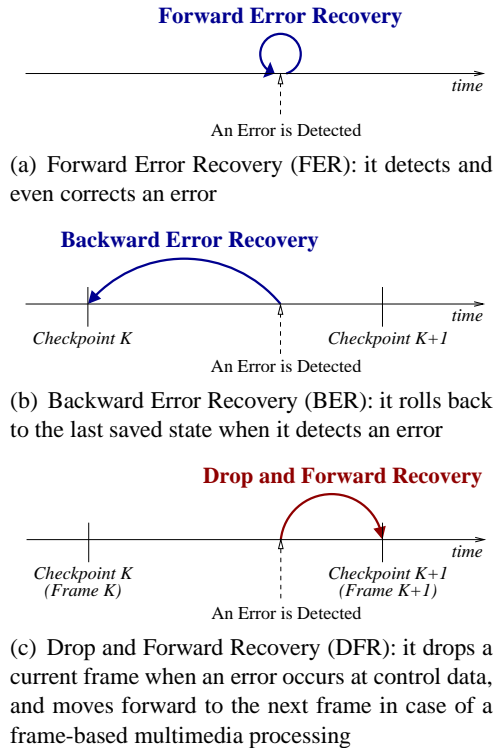


Figure 5.4: Error Recovery Mechanisms

### 5.3.3 Error-Resilient Video Encoding at Application

Error-resilient video encoding techniques have been developed to reduce the impact of transmission errors, e.g., packet losses, on the video quality [16, 54, 123]. The PBPAIR (Probability-Based Power Aware Intra Refresh) technique [54] addresses the tradeoff between energy-efficiency and compression-efficiency based on given knowledge of network errors. PBPAIR is designed to increase the compression efficiency, i.e., to decrease the encoded file size, at stable network status, and to decrease compression efficiency by increasing the number of intra-coded macro-blocks in the video when the packet loss rate is high. Inherently, PBPAIR is energy-efficient (especially when packet loss rates are higher) and adaptive (since its resilience can be adjusted for various packet loss rates). PBPAIR takes two parameters – *Packet Loss Rate (PLR)* and *Intra\_Threshold*. PLR indicates the anticipated error rate in the network and *Intra\_Threshold* can be adjusted given the user expectation of the quality. To make use of PBPAIR, our cross-layer approach converts SER for PLR, and selects *Intra\_Threshold* using the original method in

PBPAIR. We present a simple conversion for SER. First, the number of soft errors,  $N_{SE}$ , during the execution of one frame encoding is calculated as  $N_{SE} = S_{cache} \times N_{inst} \times R_{SE}$  where  $S_{cache}$  is the size of a cache in KB,  $N_{inst}$  is the number of instructions for one frame encoding, and  $R_{SE}$  is an SER per instruction per KB.  $N_{SE}$  value is then converted to a percent value and used as a FLR (Frame Loss Rate) in our study. For example, if  $S_{cache}$  is 32,  $N_{inst}$  is  $10^8$ , and  $R_{SE}$  is  $10^{-10}$ , then  $N_{SE}$  becomes 0.32. So FLR is 32%. Note that FLR becomes 100% if  $N_{SE}$  is larger than 1. Now, PBPAIR can generate the compressed video data resilient against the packet losses in networks (PLR) as well as against the soft errors at the hardware layer (FLR) as shown in Figure 5.3.

## 5.4 Intelligent Selective Algorithms

In a naive DFR approach, any single soft error at the hardware layer causes a frame drop whenever it occurs at the control data (non-multimedia data). However, naive DFR can significantly degrade the quality in case of consecutive frame drops. To prevent this result, we present a family of intelligent schemes to select a policy that balances DFR and BER based on the useful information at the mobile device.

---

```

JOINT_DFR/BER(Algo)
01: policy = DFR
02: switch (Algo)
03:   case SLACK : //Slack-Aware DFR/BER
04:      $T_{elapsed} = T_{Error} - T_K$ 
05:      $T_{threshold} = S \times T_{ACET}$ 
06:     if ( $T_{elapsed} < T_{threshold}$ )
07:       policy = BER
08:     endIf
09:   break;
10:   case FRAME : //Frame-Aware DFR/BER
11:      $F_{importance} = estimateImportance()$ 
12:     if ( $F_{importance} > F_{threshold}$ )
13:       policy = BER
14:     endIf
15:   break;
16:   case QoS : //QoS-Aware DFR/BER
17:      $Q_{current} = calcPSNR()$ 
18:     if ( $Q_{current} < Q_{threshold}$ )
19:       policy = BER
20:     endIf
21:   break;
22: endSwitch
23: return policy

```

---

Figure 5.5: Intelligent Selective Schemes

### Slack-Aware DFR/BER

The main problem with BER is its inability to guarantee delivery of multimedia service in real-time. However, if the remaining time to reach the deadline is enough to re-encode the video frame when an error is detected, we can apply BER rather than DFR for the quality improvement. Since the encoding time is varying from frame to frame and it is hard to measure the remaining time accurately, our scheme presents a knob to select a policy based on the elapsed time with ACET (Average Case Execution Time) as shown in Figure 5.5. Our knob,  $S$ , indicates the portion of ACET,  $T_{ACET}$ , that the system can endure. Thus, SA-DFR/BER (Slack-Aware DFR/BER) selects BER (Lines 04-08) as shown in Figure 5.5 if the elapsed time from the starting of the frame  $K$  encoding to the time of error occurrence,  $T_{elapsed} = T_{error} - T_K$ , is smaller than given threshold time,  $T_{threshold} = S \times T_{ACET}$ . Otherwise, SA-DFR/BER selects DFR. For example, if  $S = 0.2$  and  $T_{ACET} = 100,000$  cycles,  $T_{threshold}$  becomes 20,000 cycles. Thus, an error occurring before 20,000 cycles from the starting of the current frame encoding results in BER. The higher  $S$  value increases the probability of BER policy to be selected, and thus improves the video quality while incurring more performance and power overheads due to rolling backward recovery. Indeed, the infinite value of  $S$  always results in a BER policy and the zero value of  $S$  does a DFR policy.

### Frame-Aware DFR/BER

Each frame has a different impact on the video quality. For example, I-frames are considered more important than P-frames with the perspective of the video quality [16, 54]. Thus, if a frame in which a soft error is detected is important in terms of the video quality, FA-DFR/BER (Frame-Aware DFR/BER) rolls back and encodes this frame again (BER) to minimize the quality loss (Lines 11-14) as shown in Figure 5.5. Otherwise, it drops the current frame and moves forward to the next frame (DFR). Based on available information at the mobile device, the importance of a frame can be decided in several ways. The frame type such as I-frame or P-frame is one example, and any I-frame will be encoded eventually until no soft error is detected. Another information such as recovery history, e.g., whether the previous frame has been dropped or not due to a soft error, can be used to decide a policy. If the previous frame was dropped, FA-DFR/BER prevents the current frame from being dropped since the consecutive frame drops may degrade the video quality significantly. Also, the difference between two consecutive frames can be used to estimate the importance of a frame in terms of the video quality. The intuition behind

this approach is that the larger difference between two frames indicates the higher impact on the video quality if the current frame is lost. Thus, if the difference between them is larger than given threshold value FA-DFR/BER selects BER. Otherwise, DFR is selected. Our approach exploits the difference between consecutive frames to determine the importance of frames.

### **QoS-Aware DFR/BER**

The potential problem with a DFR mechanism is the significant degradation of the QoS due to several frame drops. Thus, QA-DFR/BER (QoS-Aware DFR/BER) selects BER when the delivered QoS is unsatisfactory with the QoS requirement. The current quality value,  $Q_{current}$ , for frames that have been encoded so far can be calculated at the end of encoding of each frame. QA-DFR/BER selects BER for the erroneous frame if  $Q_{current}$  is worse than  $Q_{threshold}$ , given threshold QoS value (Lines 17-20) as shown in Figure 5.5. Otherwise, the default policy, DFR, is selected. The delivered QoS to the decoding end is also important but this approach considering transmission errors is a future work of this thesis.

## **5.5 Effectiveness of CC-PROTECT**

### **5.5.1 Experimental Setup**

#### **5.5.1.1 System Compositions**

To demonstrate the effectiveness of our cooperative, cross-layer scheme to combat soft errors, we develop 5 system compositions, shown in Table 5.2:

1. **BASE:** This is the default composition, which does not provide any error detection and/or correction. In this composition, we use GOP (Group of Picture) video encoding [16]. For GOP, the first frame is encoded as an I-frame and the other frames are encoded as P-frames, and the quantization scale is set to 10. The middleware and operating system are unaware of soft errors, and hardware has a unified unprotected cache. BASE composition does not incur overheads for protection in terms of power and performance, but suffers from high failure rates and low multimedia quality due to no protection on internal data from hardware defects.
2. **HW-PROTECT:** In this composition, all error detection and correction are provided in hardware. This is implemented through the use of Error Correction Code (ECC) in Partially

Table 5.2: System Compositions - Our CC-PROTECT is a middleware-driven, cooperative approach aware of hardware failures

	<b>System Compositions with respect to Error Resilience</b>				
<i>Abstraction Layers</i>	BASE	HW-PROTECT	APP-PROTECT	MULTI-PROTECT	CC-PROTECT
<i>Application</i>	GOP <i>(error-prone)</i>	GOP <i>(error-prone)</i>	PBPAIR <i>(error-resilient)</i>	PBPAIR <i>(error-resilient)</i>	PBPAIR <i>(error-resilient)</i>
<i>Middleware</i>	None	None	○Monitor network errors & Inform PBPAIR of PLR	○Monitor network errors & Inform PBPAIR of PLR	○Monitor network errors & Inform PBPAIR of PLR ● <b>Translate SER</b> ● <b>Trigger Selective DFR</b> <b>(Drive cache update &amp; Inform PBPAIR)</b>
<i>Operating System</i>	None	○Map pages to a PPC	None	○Map pages to a PPC	○Map pages to a PPC ● <b>Monitor soft errors</b>
<i>Hardware</i>	Unprotected Cache <i>(error-prone)</i>	PPC with ECC <i>(error-protected)</i>	Unprotected Cache <i>(error-prone)</i>	PPC with ECC <i>(error-protected)</i>	PPC with “EDC” <i>(error-protected)</i>

Protected Caches [65]. As compared to protecting the whole cache, PPCs provide efficient reliability by just protecting the non-multimedia data (control data) against soft errors. This composition presents a low failure rate and high QoS, as it protects at the hardware level. However, it incurs overheads in terms of power and performance due to an ECC scheme.

3. **APP-PROTECT:** In this composition, all error detection and correction are provided in the application. For this, we use error-resilient video encoding PBPAIR [54]. We set the PLR parameter in PBPAIR to 0% to isolate the effects of soft errors from those of network packet losses. Intra\_Threshold is selected through the original method of PBPAIR to generate the similar size of the compressed video as GOP to ensure a fair comparison with respect to the transmission cost.
4. **MULTI-PROTECT:** In this composition, error correction is provided at all levels. We use error-resilient video encoding (PBPAIR) and a protected cache (a PPC with an ECC scheme). It implements both error-resilience at the application layer and an ECC scheme at the hardware layer.
5. **CC-PROTECT:** This is our proposed composition, in which we exploit error-resilient video encoding PBPAIR and PPC with an EDC scheme, and it supports middleware-driven mechanisms aware of soft errors such as translating SER for PBPAIR and triggering a hybrid scheme of DFR and BER.

Within our proposed composition, we study various selective schemes such as:

- **Naive DFR** - always triggers a DFR mechanism.
- **Naive BER** - always triggers a BER mechanism.
- **No DFR/BER** - never triggers a DFR or BER mechanism.
- **Random DFR/BER** - randomly triggers a DFR or BER mechanism.
- **SA-DFR/BER** - selects a DFR or BER mechanism depending on slack.
- **FA-DFR/BER** - selects a DFR or BER mechanism depending on frame.
- **QA-DFR/BER** - selects a DFR or BER mechanism depending on QoS.

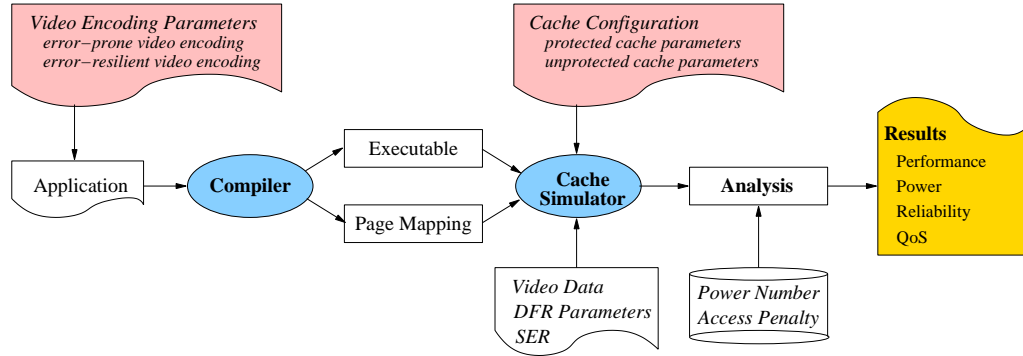


Figure 5.6: Experimental Setup – Compiler/Simulator/Analyzer

### 5.5.1.2 Simulation Setup

We perform our study on an extensive simulation environment that we have built to model the HP iPAQ h5555 [40] like processor-memory system. We have modified the *sim-cache* simulator from the SimpleScalar toolchain [11] to model the PPC architecture and to inject soft errors as in our previous work [65]. To support the unequal data protection for a PPC architecture, a compiler as shown in Figure 5.6 generates not only an executable but also a page mapping table. A page mapping table has a list of the marked global variables (multimedia data), which will be mapped into an unprotected data cache and the other data will be exclusively mapped into a protected data cache in a PPC during simulations. Note that all data will be mapped into an unprotected cache in case of an error-prone data cache.

As test video streams, *AKIYO*, *FOREMAN*, and *COASTGUARD* in QCIF format ( $176 \times 144$  pixels) are used for our simulation study, and each of them represents a video clip of low activity, medium activity, and high activity, respectively. To evaluate the cycle accurate results within reasonable amount of simulation time, 300 frames of each video stream are chopped into 75 sequences of four frames (several hours to simulate a video encoding with 300 frames of video on Sun Sparc at 1.5 GHz). We then ran a simulation at least four times with each sequence, resulting in more than 300 runs studied ( $300 \text{ runs} = 4 \text{ times of run} \times 75 \text{ sequences}$ ). DFR parameters are inputs for selective DFR/BER schemes. For instance, a slack value ( $S$ ) is given for SA-DFR/BER in Section 5.4.

The simulator models soft errors by randomly injecting single-bit errors and double-



bit errors in an unprotected data cache according to SERs. Thus, a single-bit in a data cache is randomly chosen, and a bit value at this single-bit is inverted if a randomly generated number is less than SER when an instruction is executed in the simulator. Similarly, double-bit errors are injected. Since a protected data cache is resilient against single-bit errors, only double-bit errors occur. To accomplish the experiments in a reasonable amount of time, accelerated SERs are used. SER is set to  $10^{-11}$  per KB per instruction for single-bit errors. Note that SER for current technology (about  $2.28 \times 10^{-17}$  at  $90nm$ )<sup>1</sup> is much less than this accelerated SER by several orders of magnitude, but it increases exponentially as technology scales [8, 38, 77, 124]. However, we maintain the accurate rate (about  $10^{-2}$ ) between single-bit SER and double-bit SER, thus  $10^{-13}$  per KB per instruction is used for double-bit errors.

### 5.5.1.3 Evaluation Metrics

Our simulator returns the number of accesses and the number of misses to each cache configuration. We analyzed these statistics with given power and performance numbers, and estimated access time and energy consumption of the memory subsystem as shown in Figure 5.6. QoS is measured in PSNR (Peak Signal to Noise Ratio) using the encoded video output and the original video input.

**Performance Model:** For performance evaluation of each composition, we estimate the access latency to the memory subsystem. The access latency of memory subsystem  $L$  is estimated as  $L = (A_{cache} \times L_{access}) + (M_{cache} \times L_{miss}) + (N_{policy} \times L_{policy})$  where  $A_{cache}$  is the number of accesses to a cache,  $L_{access}$  is the cache access time,  $M_{cache}$  is the number of misses to a cache,  $L_{miss}$  is the cache miss penalty, i.e., the access penalty to a bus and a memory,  $N_{policy}$  is the number of triggered policies such as DFR and BER, and  $L_{policy}$  is the latency penalty for a policy. The overhead of delay for ECC is estimated and synthesized using the CACTI [106] and the Synopsys Design Compiler [113] as in our previous work [65], and the overhead of delay for EDC is calculated using the ratio between delays of ECC and EDC from previous papers [65, 72]. Also, the delay overheads for DFR and BER are estimated through the simulations so that the overheads for context switch and checkpoints are added at the analysis stage in our simulation study as shown

---

<sup>1</sup>It is projected using an exponentially increasing ratio from 1,000 FIT/Mbit at  $180nm$  technology to 100,000 FIT/Mbit at  $130nm$  technology [8, 77].

in Figure 5.6.

**Energy Model:** We estimate the energy consumption of the memory subsystem using the power models presented in our previous work [108]. The overheads of power for a Hamming code (38,32) and a parity code are synthesized and estimated similar to those for delay. The power consumption penalty for a recovery policy such as DFR and BER is estimated through simulations. The energy consumption of the memory subsystem  $E$  is computed as  $E = (A_{cache} \times P_{access}) + (M_{cache} \times P_{miss}) + (N_{policy} \times P_{policy})$  where  $P_{access}$  is the power consumption per cache access,  $P_{miss}$  is the power penalty per cache miss, and  $P_{policy}$  is the power penalty for a recovery policy. Due to the lack of space, power and delay penalties are detailed in our technical report [68].

**Failure Rate Model:** To estimate reliability, we define an execution as a *Success* if it ends within twice the normal execution time and returns the correct output opened by a decoder. Otherwise, we assume that it is a *Failure* due to causes such as a system crash, infinite loop, or segmentation fault. Note that the degradation of video data is not considered as a failure in our study. The failure rate has been obtained through at least hundreds of executions for each composition by counting the number of failures out of a total number of executions based on the binomial distribution analysis [68].

**Quality of Service Model:** We estimate the QoS in PSNR. PSNR is defined in dB as  $PSNR = 10 \log_{10}(\frac{MAX^2}{MSE})$ , where  $MAX$  is the maximum pixel value and  $MSE$  is the Mean Squared Error, which is the mean of the square of differences between the pixel values of the erroneous video output (due to soft errors and frame drops), and of the correctly reconstructed output (without errors).

## 5.5.2 Experimental Results

We present two sets of experiments. First, we demonstrate the effectiveness of our cooperative, cross-layer methods in low-cost reliability at a slight degradation of QoS for different video streams (Section 5.5.2.1). Second, we show the effectiveness of intelligent DFR/BER selection schemes to improve the video quality (Section 5.5.2.2).

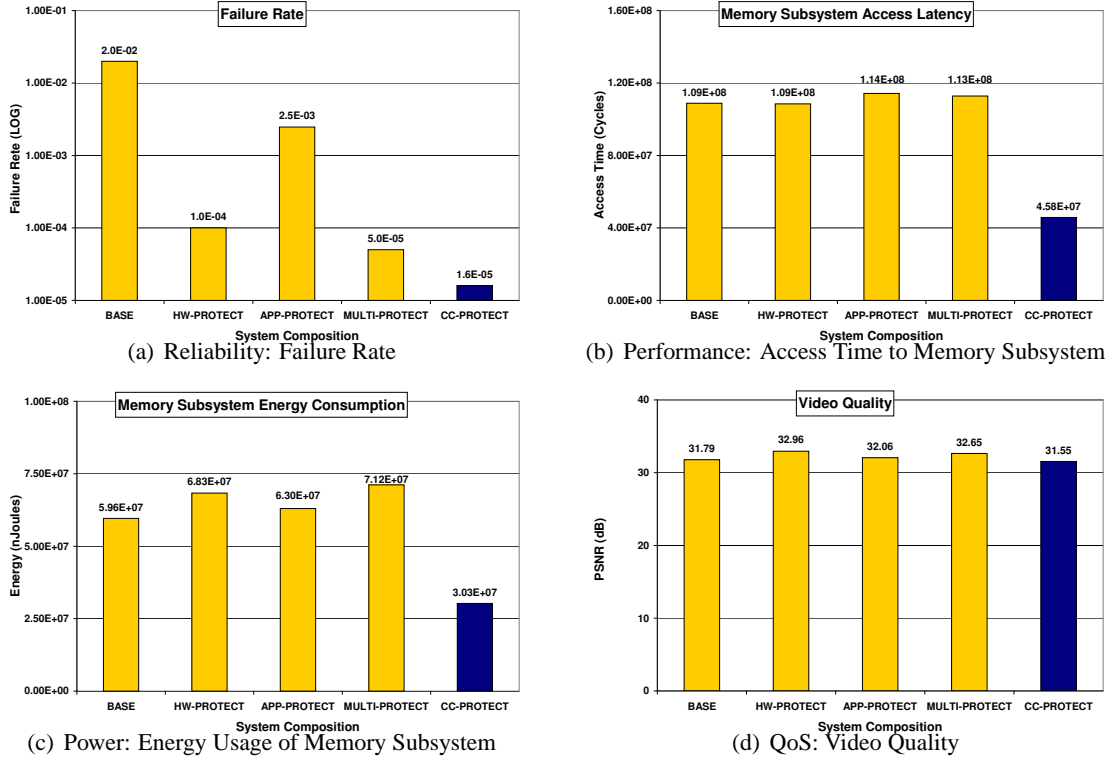


Figure 5.7: CC-PROTECT achieves the low-cost reliability at the minimal QoS degradation

### 5.5.2.1 Effectiveness of CC-PROTECT

Figure 5.7 clearly shows that our cross-layer, error-aware approach increases the reliability with the minimal costs of performance and energy consumption as well as the minimal degradation of video quality.

Figure 5.7(a) clearly demonstrates that our CC-PROTECT (PBPAIR, a DFR mechanism, and a PPC with an EDC protection) improves the failure rate by more than 1,000 times than that of BASE (GOP and an unprotected data cache). This reliability improvement is mainly due to the coupling of the error detection and a DFR mechanism in a cross-layered manner. While HW-PROTECT and MULTI-PROTECT have lower failure rates than that of BASE, CC-PROTECT has lower failure rate than either of them. This is because it has less time to be exposed to soft errors due to the combined effects of a frame drop and the performance efficiency of PBPAIR over GOP. It is important that APP-PROTECT (PBPAIR and an unprotected data cache) shows the failure rate close to that of BASE since a failure results from errors on control data, which are




not protected in APP-PROTECT. Thus, our CC-PROTECT can achieve the best reliability among all compositions.

Figure 5.7(b) shows that our CC-PROTECT scheme is the best in terms of performance. It reduces the memory subsystem access time by 58%, compared to that of BASE. It is very effective since CC-PROTECT reduces the failure rate by 1,000 times and it reduces the access latency of memory subsystem compared to BASE. Note that all the other compositions incur a performance overhead but CC-PROTECT improves the performance. This performance improvement is a result of skipping intensive compression algorithms due to a DFR mechanism and the performance efficiency of PBPAIR algorithms. However, the performance efficiency of PBPAIR is not well exploited in case of APP-PROTECT, (showing 5% overhead compared to BASE) because PBPAIR increases the compression efficiency rather than the performance efficiency at low PLR such as 0% PLR. For the same reason, MULTI-PROTECT incurs about 4% overhead compared to BASE. Indeed, HW-PROTECT does not incur the performance overhead because a PPC achieves high performance by protecting only non-multimedia data [65].

From the perspective of energy consumption of the memory subsystem, our CC-PROTECT scheme saves energy consumption by 49%, 56%, 52%, and 57% as compared to BASE, HW-PROTECT, APP-PROTECT, and MULTI-PROTECT, respectively, as shown in Figure 5.7(c). CC-PROTECT reduces the energy consumption of memory subsystem due to (i) less expensive EDC technique than ECC, (ii) skipping expensive compression algorithms due to a cooperative DFR mechanism, and (iii) energy efficiency of PBPAIR by introducing more intra-coded macro-blocks than expensive inter-coded macro-blocks. Note that all other compositions incur overheads of performance and power compared to BASE except for CC-PROTECT. Thus, CC-PROTECT can even reduce the power and access time of memory subsystem while obtaining high reliability.

Our CC-PROTECT scheme achieves video quality close to those of other compositions as shown in Figure 5.7(d). While an EDC scheme protects the non-multimedia data in CC-PROTECT, a frame drop due to a DFR mechanism degrades the video quality. Note that PBPAIR algorithms can improve this video quality by increasing the resilience level at the cost of the compressed video size (causing the transmission costs of power and delay). However, CC-PROTECT saves at least 49% of power and performance for the minimal failure rate at the minimal cost of QoS by up to 1.41 dB (less than 5% quality degradation) compared to all the compositions. Note

Table 5.3: CC-PROTECT is very effective in terms of performance, power, and reliability at the minimal QoS degradation for different video streams (normalized result of each composition to that of BASE)

Video Stream	System Composition	Access Time	Energy Consumption	Failure Rate	Video Quality
<i>AKIYO</i> <i>(low activity)</i> 	BASE	1	1	1	1
	HW-PROTECT	0.99	1.14	0.4 E-2	1.02
	APP-PROTECT	0.87	0.89	13.2 E-2	1.01
	MULTI-PROTECT	0.89	1.03	0.2 E-2	1.02
	<b>CC-PROTECT</b>	<b>0.27</b>	<b>0.34</b>	<b>0.1 E-2</b>	<b>1.02</b>
<i>FOREMAN</i> <i>(medium activity)</i> 	BASE	1	1	1	1
	HW-PROTECT	1	1.15	0.5 E-2	1.04
	APP-PROTECT	1.05	1.06	12.3 E-2	1.01
	MULTI-PROTECT	1.04	1.19	0.3 E-2	1.03
	<b>CC-PROTECT</b>	<b>0.42</b>	<b>0.51</b>	<b>0.1 E-2</b>	<b>0.99</b>
<i>COASTGUARD</i> <i>(high activity)</i> 	BASE	1	1	1	1
	HW-PROTECT	0.99	1.14	0.4 E-2	1.03
	APP-PROTECT	1.09	1.1	13.0 E-2	0.99
	MULTI-PROTECT	1.06	1.23	0.3 E-2	1.02
	<b>CC-PROTECT</b>	<b>0.49</b>	<b>0.58</b>	<b>0.1 E-2</b>	<b>0.93</b>

that these results come from only one soft error at the protected cache in a PPC, and the video quality may degrade significantly due to multiple frame drops resulting from multiple occurrences of soft errors. We present the experimental results for those cases in Section 5.5.2.2.

Table 5.3 summarizes the normalized results of each composition to those of BASE in terms of performance, power, reliability, and QoS for different video streams. This table clearly shows that CC-PROTECT has the least costs of power and performance for the minimal failure rate with the minimal QoS degradation for all video streams. The interesting observation that we can make from this table is that we can even improve the video quality while still saving the performance and power costs (73% and 66%, respectively) compared to BASE for a video stream *AKIYO*. This quality improvement (about 2%) is because: (i) a frame drop may not affect the video quality for a video stream with low-activity such as *AKIYO* and (ii) less amount of execution time of PBPAIR results in less exposure of a data cache to soft errors. Indeed, the QoS impact of one frame drop for *AKIYO* is about 0.08% on average. On the other hand, for high activity of video stream such as *COASTGUARD*, our CC-PROTECT degrades the video quality by about 6% in PSNR. But still CC-PROTECT demonstrates the least access time and energy consumption for the minimal failure rate. Note that all these results are evaluated under the condition of no errors in the network. We also observed similar results under various network status configurations. For example, at 10% PLR, our CC-PROTECT reduces access time of memory subsystem by 58%, while APP-PROTECT saves it by 32% (more error rate triggers more intra-coded macro-

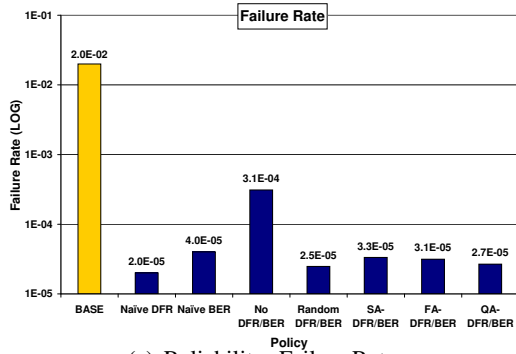
blocks, causing high performance in PBPAIR algorithms), as compared to BASE. We have also run simulations for different compositions and similar results demonstrated the effectiveness of our CC-PROTECT. For instance, a composition (GOP and a 32 KB of protected cache with an ECC – forward error recovery) incurs 45% performance and 34% energy overheads as compared to BASE. This is because all data (multimedia data and control data) are protected from soft errors with an expensive ECC scheme. Due to the lack of space, more results are available in our technical report [68].

In summary, our cooperative, cross-layer methods exploit a DFR mechanism with an inexpensive EDC protection to decrease the failure rate by about  $1,000\times$ , and an error-resilient video encoding technique to minimize the quality degradation by 2% while significantly saving the access time by 61% and energy consumption by 52% on average over multiple video streams, as compared to BASE. Also, our cooperative, cross-layer approach achieves a better reliability than a previously proposed PPC architecture with an ECC protection at the cost of 3% QoS degradation while reducing the access time by 60% and the energy consumption by 58%.

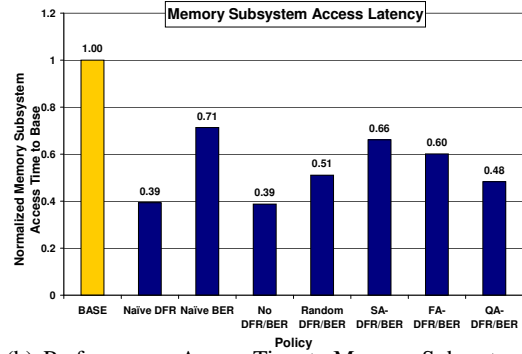
### 5.5.2.2 Effectiveness of Intelligent Selective Schemes

Our CC-PROTECT scheme outperforms all possible compositions in terms of performance, power, and reliability while it slightly degrades the video quality mainly due to frame drops when soft errors occur. Figure 5.8 demonstrates that all intelligent selective schemes improve the video quality without incurring performance and energy costs significantly (still mostly lower than costs of BASE).

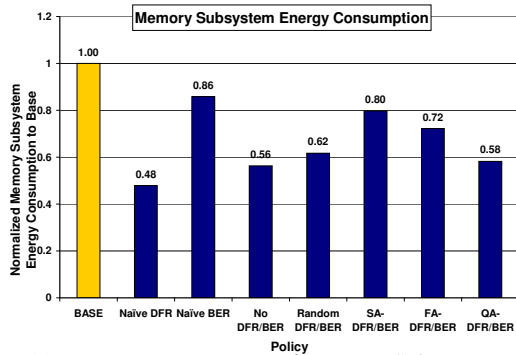
In Figure 5.8, the X-axis represents selective mechanisms compared to BASE. Note that they are all running PBPAIR on a PPC architecture with an EDC scheme except for BASE (running GOP on an unprotected cache) and No DFR/BER (running PBPAIR on a PPC without any protection) for comparison. We parameterize a policy selection based on available information in a mobile device. The Naive DFR scheme shows the worst video quality as shown in Figure 5.8(d) at the least costs in terms of power and performance as shown in Figure 5.8(b) and Figure 5.8(c). Note that Naive DFR in Figure 5.8 results from multiple soft errors (1.7 errors on average) on the protected data cache in a PPC, which degrades the video quality worse than that of CC-PROTECT in Figure 5.7(d). On the other hand, Naive BER scheme presents better video quality than that of



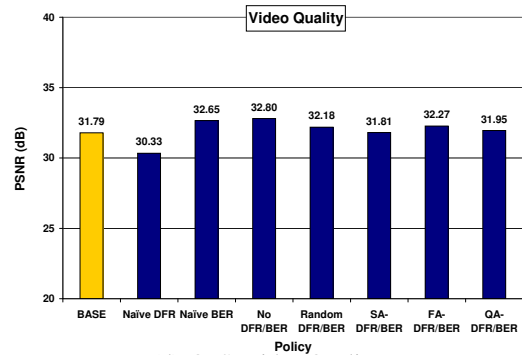
(a) Reliability: Failure Rate



(b) Performance: Access Time to Memory Subsystem



(c) Power: Energy Usage of Memory Subsystem



(d) QoS: Video Quality

Figure 5.8: Intelligent selective schemes maintain the video quality and reliability with minimal overheads of power and performance

Naive DFR while incurring the most expensive power and performance costs compared to other schemes. In terms of reliability, Naive BER shows worse failure rate than that of Naive DFR as shown in Figure 5.8(a). This is mainly because Naive BER increases the execution time, causing more time for a PPC to be exposed to soft errors. Clearly, No DFR/BER does not have a mechanism to protect a system from soft errors, causing very high failure rate as shown in Figure 5.8(a). Note that Figure 5.8(d) shows higher video quality of No DFR/BER than others. This is because we measured the video quality in PSNR when simulations are successes where No DFR/BER does not skip any frame. Random DFR/BER provides good video quality with inexpensive power and performance. For SA-DFR/BER, the results have been profiled with the knob  $S$  (the portion of ACET) from 0% to 100% in 10% increments, and SA-DFR/BER with  $S = 60\%$  is compared in Figure 5.8 since it is the least value of the knob to recover the video quality better than that of BASE according to profiled results. However, it is an expensive approach since it incurs high

overheads in terms of power and performance while it presents a better video quality than that of Naive DFR. For FA-DFR/BER scheme, our preliminary experiments show that the difference in PSNR between consecutive frames make the 2<sup>nd</sup>, 3<sup>rd</sup>, and 4<sup>th</sup> frame in the descending order of the importance on average. Thus, FA-DFR/BER with 2<sup>nd</sup> frame is studied to improve the video quality most and indicates that we select BER rather than DFR whenever a soft error occurs in encoding the 2<sup>nd</sup> frame. In these particular experiments, FA-DFR/BER scheme is more effective than SA-DFR/BER scheme since it has lower costs with better QoS (failure rates are close). For QA-DFR/BER, 31.79 dB is considered as the QoS threshold value since it is the average video quality in case of BASE. QA-DFR/BER provides lower video quality while incurring less costs than FA-DFR/BER scheme. Thus, each selective scheme has pros and cons in terms of performance, power, reliability, and QoS.

In summary, selective DFR/BER mechanisms allow a system to maintain the video quality and reliability with minimal costs of power and performance.

## 5.6 Summary

Reliability is of paramount concern in mobile devices where the resources such as power and performance are constrained. In order to resolve the complexity of trade-offs among multi-dimensional properties, a cross-layer approach from the hardware layer to the application layer should be taken into account since traditional techniques are unable to address the impacts of an approach on other properties at other layers, and are unable to drive the whole system's reliability in a power and performance efficient manner.

Traditionally, reliability techniques have been developed at individual levels, and have remained seemingly incognizant of the strategies employed at other levels. While focusing their attention to a single level, researchers make a general assumption that no other schemes are operational at other levels. We believe that the cumulative effect of reliability schemes at multiple levels can be potentially significant; but this also requires careful evaluation of the trade-offs involved and the customizations required for unified operation.

In this chapter, we present our cross-layer strategy for low-cost reliability, CC-PROTECT or Cooperative, Cross-layer Protection. By synergistic cooperation among PPC with EDC at the



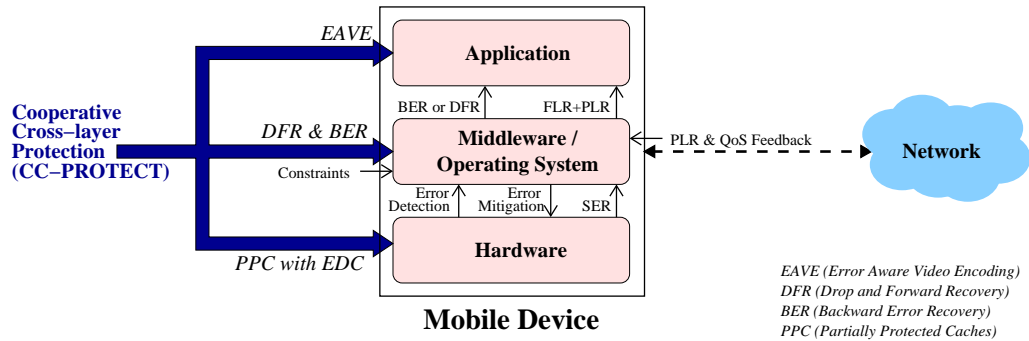


Figure 5.9: Cooperative Cross-Layer Protection – CC-PROTECT exploits existing error control schemes across system abstraction layers to achieve cost efficient reliability

hardware level, DFR and BER at the middleware level, and EAVE at the application level on mobile multimedia systems as shown in Figure 5.9, we obtain high reliability, high performance, and high energy saving at the cost of slight QoS degradation. With the perspective of reliability, a less expensive EDC scheme for PPC architectures than an ECC scheme can detect an error, and a DFR mechanism recovers an erroneous state by dropping a current encoding frame and moving forward to the next frame encoding in mobile video applications as shown in Figure 5.9. With the perspective of QoS, CC-PROTECT exploits an error-aware video encoding to reduce the impact of a frame drop due to DFR by translating SER into FLR, and considering a frame drop as a frame loss due to packet losses in networks as shown in Figure 5.9.

While traditional protection techniques deploy compositions of error-resilient techniques or protected schemes without cooperation in a cross-layered manner, our CC-PROTECT scheme has been demonstrated as a very effective method to improve the performance and energy consumption rather than incurring overheads with better reliability at the minimal cost of QoS degradation. CC-PROTECT also expands the tradeoff space for multi-dimensional design constraints, and the applicability of existing error control schemes across system abstraction layers.

# Chapter 6

## Conclusion

### 6.1 Summary

With rapid advancement of technology and wide deployment of wireless communication, multimedia applications are becoming popular in mobile embedded systems. On the other hand, reliability is becoming a serious concern for embedded system design due to high integration of complex algorithms and increasing error rates as technology scales. For example, soft error rates are expected to increase exponentially with each technology in SRAM architectures. It is challenging to achieve low-cost reliability since the resources are significantly constrained in mobile embedded systems. Further, there exist tradeoffs among these design constraints. For example, conventional redundancy techniques such as error correction codes incur high overheads in terms of performance, power, and area cost while voltage scaling techniques increase the soft error rate exponentially.

This thesis proposed a cooperative, cross-layer methodology to design reliable mobile embedded systems with minimal costs. Previously, cross-layer approaches have been focused on tradeoffs among performance, power, timing, and QoS, but not reliability together. We have presented PPC (Partially Protected Caches) architectures, EAVE (Error-Aware Video Encoding), and CC-PROTECT (Cooperative, Cross-layer Protection) as cooperative cross-layer strategies in this thesis:

- **PPC** is a cross-layer strategy for error-resilient microarchitecture at the hardware layer with minimal overheads in terms of power, performance, and area by exploiting the natural error-tolerance of multimedia data and the vulnerable time of data and codes at the application

layer.

- **EAVE** is a cross-layer strategy for maximal energy reduction by actively exploiting errors at the middleware layer and video encoding at the application layer, which was originally developed to encode video data resilient against network errors in an energy-efficient way.
- **CC-PROTECT** is a cross-layer strategy to achieve low-cost reliability for mobile multimedia embedded systems by exploiting PPC architectures with an error detection code at the hardware layer, developing a joint recovery scheme at the middleware layer, and extending the applicability of EAVE for hardware-induced frame drops at the application layer.

## 6.2 Contribution

The contributions and results of cross-layer strategies that this thesis presents are enumerated below:

1. This thesis develops cooperative and cross-layer methods for low-cost reliability:
  - PPC strategy includes the hardware and application level schemes [65, 64, 67, 47, 66].
  - EAVE strategy includes the application, middleware, and network level schemes [63, 48].
  - CC-PROTECT strategy includes the application, middleware, operating system, and hardware level schemes [69, 68].
2. This thesis significantly expands the tradeoff space considering multiple design constraints such as performance, energy consumption, reliability, and QoS.
3. This thesis saves resources significantly with minimal costs.
  - PPC strategy improves the performance by 16% and the energy consumption by 8% at the cost of QoS degradation, as compared to a previously proposed cache with an error correction protection [67].
  - EAVE strategy reduces the energy consumption by 37% without QoS degradation, as compared to a normal video encoding [63].

- CC-PROTECT strategy improves the performance by 60%, the energy consumption by 58% at the minimal QoS degradation, as compared to a previously proposed hardware-based protection [69].
4. This thesis extends the applicability of existing techniques.
- PPC strategy extends the previously proposed HPC (Horizontally Protected Caches) architecture for the purpose of reliability.
  - EAVE strategy extends the applicability of error-resilient video encodings for active error exploitation, i.e., intentional frame dropping.
  - CC-PROTECT strategy extends an error-resilient video encoding and a drop and forward recovery technique against hardware defects.

### 6.3 Future Directions

The future work of this thesis includes the expansion of our strategies for different classes of errors across system abstraction layers in mobile embedded systems. For example, CC-PROTECT presents a simple method to translate the soft error rate at the protected cache into the frame loss rate at the middleware layer, and we plan to extend the translation and integration methods for different types of errors across system layers within embedded systems as well as in distributed embedded systems. By presenting a method for error rate conversion, CC-PROTECT can be applied for various classes of errors in a resource-efficient way. Also, we plan to apply our cooperative, cross-layer strategies for different components. For instance, our CC-PROTECT scheme can be used not only for cache protection but also for logic components against temporary faults in mobile embedded systems.

The methodology of this thesis can be expanded in the context of distributed embedded systems. In particular, we plan to extend our strategies considering dynamic network status as well as environmental contexts. For example, our presented knobs such as the error injection rate in EAVE and threshold values for intelligent selective schemes in CC-PROTECT can be adjusted according to current network status and categories of remaining resources for each mobile device attending a session in distributed system environments. Also, pervasive computing environments

can provide important contexts, and they will be involved to determine our policies that extend strategies outlined in this thesis.

# Bibliography

- [1] Sherif Abdelwahed, Nagarajan Kandasamy, and Sandeep Neema. Online control for self-management in computing systems. In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 368–375, May 2004.
- [2] Lars Åke Larzon, Ulf Bodin, and Olov Schelén. Hints and notifications. In *Proceedings of IEEE Wireless Communication and Network Conference*, pages 635–641, March 2002.
- [3] Ian F. Akyildiz and Xudong Wang. Cross-layer design in wireless mesh networks. *57(2):1061–1076*, Mar. 2008.
- [4] L. Anghel and M. Nicolaidis. Cost reduction and evaluation of a temporary faults detecting technique. In *IEEE/ACM Design Automation and Test in Europe Conference (DATE)*, pages 591–597, 2000.
- [5] Ghazanfar-Hossein Asadi, Vilas Sridharan, Mehdi B. Tahoori, and David Kaeli. Balancing performance and reliability in the memory hierarchy. In *ISPASS*, 2005.
- [6] Hossein Asadi, Vilas Sridharan, Mehdi B. Tahoori, and David Kaeli. Reliability Tradeoffs in Design of Cache Memories. In *Workshop on Architectural Reliability in conjunction with MICRO*, 2005.
- [7] Ivan V. Bajic. Efficient cross-layer error control for wireless video multicast. *53(1):276–285*, Mar 2007.
- [8] Robert Baumann. Soft errors in advanced computer systems. *IEEE Design and Test of Computers*, pages 258–266, 2005.

- [9] Mark P. Baze, Steven P. Buchner, and Dale McMorrow. A digital CMOS design technique for SEU hardening. *IEEE Trans. on Nuclear Science*, 47(6):2603–2608, Dec 2000.
- [10] Randeep Bhatia and Murali Kodialam. On power efficient communication over multi-hop wireless networks: Joint routing, scheduling and power control. In *Proceedings of IEEE International Conference on Computer and Communications (INFOCOM)*, 2004.
- [11] Doug Burger and Todd M. Austin. The SimpleScalar Tool Set, version 2.0. *SIGARCH Computer Architecture News*, 25(3):13–25, 1997.
- [12] Amit Butala and Lang Tong. Cross-layer design for medium access control in cdma ad hoc networks. *EURASIP Journal on Applied Signal Processing*, 2005(1):129–143, 2005.
- [13] Y. Cai, M. T. Schmitz, A. Ejlali, B. M. Al-Hashimi, and S. M. Reddy. Cache size selection for performance, energy and reliability of time-constrained systems. In *ASP-DAC*, 2006.
- [14] G. Chen, M. Kandemir, M. J. Irwin, and G. Memik. Compiler-Directed Selective Data Protection Against Soft Errors. In *ASP-DAC*, 2005.
- [15] Liang Cheng and Magda El Zarki. An adaptive error resilient video encoder. In *SPIE Visual Communication and Image Processing*, July 2003.
- [16] Liang Cheng and Magda El Zarki. PGOP: An error resilient techniques for low bit rate and low latency video communications. In *Picture Coding Symposium (PCS)*, Dec 2004.
- [17] Mung Chiang. Balancing transport and physical layers in wireless multihop networks: Jointly optimal congestion control and power control. *IEEE Journal on Selected Areas in Communications*, 23(1):104–116, Jan. 2005.
- [18] Mung Chiang, Steven H. Low, A. Robert Calderbank, and Jonh C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, Jan. 2007.
- [19] R. Cornea, N. Dutt, R. Gupta, I. Krueger, A. Nicolau, D. Schmidt, and S. Shukla. FORGE: A framework for optimization of distributed embedded systems software. In *International Parallel and Distributed Processing Symposium (IPDPS)*, April 2003.

- [20] G. Dimić, N. D. Sidiropoulos, and R. Zhang. Medium access control–physical cross-layer design. *IEEE Signal Processing Magazine*, 21(5):40–50, Sep. 2004.
- [21] Fred Douglass, Ramón Cáceres, Frans Kaashoek, Kai Li, Brian Marsh, and Joshua A. Tauber. Storage alternatives for mobile computers. In *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation (OSDI)*, page 3, 1994.
- [22] Fred Douglass, P. Krishnan, and Brian N. Bershad. Adaptive disk spin-down policies for mobile computers. In *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing (MLICS)*, pages 121–137, 1995.
- [23] DYNAMO. *Power Aware Middleware for Distributed Mobile Computing*. University of California at Irvine, <http://dynamo.ics.uci.edu/>.
- [24] Y. Eisenberg, C. Luna, T. Pappas, R. Berry, and A. Katsaggelos. Joint source coding and transmission power management for energy efficient wireless video communications. *IEEE Trans. Circuits Syst. Video Technology*, 12(6):411–424, 2002.
- [25] Carla Schlatter Ellis. The case for higher-level power management. In *Proceedings of the The Seventh Workshop on Hot Topics in Operating Systems (HOTOS)*, page 162, 1999.
- [26] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, Toan Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: A low-power pipeline based on circuit-level timing speculation. In *MICRO-36*, pages 7–13, 2003.
- [27] N. Feamster and H. Balakrishnan. Packet loss recovery for streaming video. In *12th International Packet Video Workshop*, April 2002.
- [28] Jason Flinn, Eyal de Lara, Mahadev Satyanarayanan, Dan S. Wallach, and Willy Zwaenepoel. Reducing the energy usage of office applications. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg (Middleware)*, pages 252–272, 2001.
- [29] Jason Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. *SIGOPS Operating Systems Review*, 33(5):48–63, Dec. 1999.



- [30] Jason Flinn and M. Satyanarayanan. PowerScope: A tool for profiling the energy usage of mobile applications. In *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications (WMCSA)*, pages 2–10, Feb. 1999.
- [31] FORGE Project. *A Framework for Optimization of Distributed Embedded Systems Software*. University of California at Irvine, <http://www.ics.uci.edu/forge/>.
- [32] W. Ge, J. Zhang, and S. Shen. Cross-layer design approach to multicast in wireless networks. 6(3):1063–1071, Mar. 2007.
- [33] A. Gonz'alez, C. Aliagas, and M. Valero. A data cache with multiple caching strategies tuned to different types of locality. In *International Conference on Supercomputing (ICS)*, pages 338–347, July 1995.
- [34] GRACE Project. *Global Resource Adaptation through CooperAtion*. University of Illinois at Urbana-Champaign, <http://rsim.cs.uiuc.edu/grace/>.
- [35] L. Guo, X. Ding, H. Wang, Q. Li, S. Chen, and X. Zhang. Exploiting idle communication power to improve wireless network performance and energy efficiency. In *IEEE International Conference on Computer and Communications (INFOCOM)*, pages 1–12, April 2006.
- [36] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown. MiBench: A free, commercially representative embedded benchmark suite. In *IEEE Workshop Workload Characterization*, Dec 2001.
- [37] Al Harris, Cigdem Sengul, Robin Kravets, and Prashant Ratanchandani. Energy-efficient multimedia communications in lossy multi-hop wireless networks. *IFIP Mobile and Wireless Communication Networks*, 162:461–472, 2005.
- [38] P. Hazucha and C. Svensson. Impact of CMOS technology scaling on the atmospheric neutron soft error rate. *IEEE Trans. on Nuclear Science*, 47(6):2586–2594, 2000.
- [39] David P. Helmbold, Darrell D. E. Long, and Bruce Sherrod. A dynamic disk spin-down technique for mobile computing. In *Proceedings of the 2nd annual international conference on Mobile computing and networking (MobiCom)*, pages 130–142, 1996.

- [40] Hewlett Packard, <http://www.hp.com>. *HP iPAQ h5555 Series - System Specifications*.
- [41] Shaoxiong Hua, Gang Qu, and Shuvra S. Bhattacharyya. Energy reduction techniques for multimedia applications with tolerance to deadline misses. In *Proceedings of the 40th conference on Design Automation (DAC)*, pages 131–136, 2003.
- [42] Christopher J. Hughes, Jayanth Srinivasan, and Sarita V. Adve. Saving energy with architectural and frequency adaptations for multimedia applications. In *Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture (MICRO)*, pages 250–261, 2001.
- [43] Tomasz Imielinski, Monish Gupta, and Sarma Peyyeti. Energy efficient data filtering and communication in mobile wireless computing. In *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing (MLICS)*, pages 109–120, 1995.
- [44] Intel Corporation, <http://www.intel.com/design/pca/applicationsprocessors/manuals/278693.htm>. *Intel PXA255(R) Processor: Developer's Manual*.
- [45] ITU-T. H.263 Draft: Video Coding for Low Bitrate Communication. Draft ITU-T recommendation H.263, ITU-T, May 1996.
- [46] Yu Jiao and Ali R. Hurson. Adaptive power management for mobile agent-based information retrieval. In *IEEE Advanced Information Networking and Applications (AINA)*, pages 675–680, March 2005.
- [47] K. Lee, A. Shrivastava, I. Issenin, and N. Dutt, Technical Report (<http://www.ics.uci.edu/~kyoungwl/softerror/>). *Horizontally Partitioned Caches to Reduce Failures due to Soft Errors for Mission-Critical Multimedia Embedded Systems*, May 2006.
- [48] K. Lee, M. Kim, N. Dutt, and N. Venkatasubramanian, Technical Report (<http://www.ics.uci.edu/~kyoungwl/eepbpair/>). *Adaptive EE-PBPAIR: A Novel Error-Exploiting Video Encoder Incorporating End-to-End QoS Feedback*, Dec 2007.
- [49] Vikas Kawadia and P. R. Kumar. A cautionary perspective on cross-layer design. *IEEE Wireless Communications*, 12(1):3–11, Feb. 2005.

- [50] Amin Khajeh, Shih-Yang Cheng, Ahmed M. Eltawil, Fadi J. Kurdahi, and Rouwaida Kanj. Power management for cognitive radio platforms. In *Proceedings of IEEE Global Telecommunications Conference (GlobeCom)*, pages 4066–4070, Nov. 2007.
- [51] Amin Khajeh, Minyoung Kim, Nikil Dutt, Ahmed M. Eltawil, and Fadi J. Kurdahi. Cross-layer co-exploration of exploiting error resilience for video over wireless applications. In *IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia*, Oct. 2008.
- [52] S. Khan, M. Sqroi, E. Steinbach, and W. Kellerer. Cross-layer optimization for wireless video streaming – performance and cost. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, July 2005.
- [53] M. Kim, N. Dutt, N. Venkatasubramanian, and C. Talcott. xTune: Online verifiable cross-layer adaptation for distributed real-time embedded systems. *ACM SIGBED Review: Special Issue on the RTSS Forum on Deeply Embedded Real-Time Computing*, 5(1), Jan 2008.
- [54] M. Kim, H. Oh, N. Dutt, A. Nicolau, and N. Venkatasubramanian. PBPAIR: An energy-efficient error-resilient encoding using probability based power aware intra refresh. *ACM SIGMOBILE Mobile Computing and Communications Review*, 10(3):58–69, July 2006.
- [55] Minyoung Kim, Mark-Oliver Stehr, Carolyn Talcott, Nikil Dutt, and Nalini Venkatasubramanian. Constraint refinement for online verifiable cross-layer system adaptation. In *Proceedings of the conference on Design, automation and test in Europe (DATE)*, pages 646–651, April 2008.
- [56] Soontae Kim. Area-efficient error protection for caches. In *DATE'06*, pages 1282–1287, Mar 2006.
- [57] R. Kravets, K. Schwan, and K. Calvert. Power-aware communication for mobile computers. In *Proceedings of International Workshop on Mobile Multimedia Communications (MoMuC)*, 1999.
- [58] Robin Kravets and P. Krishnan. Power management techniques for mobile communication. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom)*, pages 157–168, 1998.

- [59] S. Krishnamohan and N. R. Mahapatra. An efficient error-masking technique for improving the soft-error robustness of static CMOS circuits. In *SOCC*, Sep 2004.
- [60] Dan Krueger, Erin Francom, and Jack Langsdorf. Circuit design for voltage scaling and SER immunity on a quad-core Itanium processor. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 94–95, 2008.
- [61] Fadi Kurdahi, Ahmed Eltawil, Amin K. Djahromi, Mohammad Makhzan, and Stanley Cheng. Error-aware design. In *Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD)*, pages 8–15, 2007.
- [62] Eyal De Lara, Dan S. Wallach, and Willy Zwaenepoel. Puppeteer: Component-based adaptation for mobile computing. In *Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems (USITS)*, pages 14–14, 2001.
- [63] Kyoungwoo Lee, Minyoung Kim, Nikil Dutt, and Nalini Venkatasubramanian. Error exploiting video encoder to extend energy/QoS tradeoffs for mobile embedded systems. In *6th IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES)*, Sep. 2008, to appear.
- [64] Kyoungwoo Lee, Aviral Shrivastava, Nikil Dutt, and Nalini Venkatasubramanian. Data partitioning techniques for partially protected caches to reduce soft error induced failures. In *6th IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES)*, Sep. 2008, to appear.
- [65] Kyoungwoo Lee, Aviral Shrivastava, Ilya Issenin, Nikil Dutt, and Nalini Venkatasubramanian. Mitigating soft error failures for multimedia applications by selective data protection. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, pages 411–420, Oct 2006.
- [66] Kyoungwoo Lee, Aviral Shrivastava, Ilya Issenin, Nikil Dutt, and Nalini Venkatasubramanian. Partially protected caches to reduce soft error induced failures in multimedia systems. Technical Report TR 06-03, UCI, June 2008.

- [67] Kyoungwoo Lee, Aviral Shrivastava, Ilya Issenin, Nikil Dutt, and Nalini Venkatasubramanian. Partially protected caches to reduce failures due to soft errors in multimedia applications. *IEEE Trans. on Very Large Scale Integration Systems (TVLSI)*, 2008, to appear.
- [68] Kyoungwoo Lee, Aviral Shrivastava, Minyoung Kim, Nikil Dutt, and Nalini Venkatasubramanian. Cross-layer interactions of error control schemes in mobile multimedia systems. Technical Report TR 08-09, University of California at Irvine, Jul 2008.
- [69] Kyoungwoo Lee, Aviral Shrivastava, Minyoung Kim, Nikil Dutt, and Nalini Venkatasubramanian. Mitigating the impact of hardware defects on multimedia applications – a cross-layer approach. In *Proceedings of the 16th annual ACM international conference on Multimedia*, Oct. 2008.
- [70] Jin-Fu Li and Yu-Jane Huang. An error detection and correction scheme for RAMs with partial-write function. In *IEEE International Workshop on Memory Technology, Design and Testing (MTDT)*, pages 115–120, 2005.
- [71] Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson. A quantitative analysis of disk drive power management in portable computers. In *Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference (WTEC)*, pages 22–22, 1994.
- [72] Lin Li, Vijay Degalahal, N. Vijaykrishnan, Mahmut Kandemir, and Mary Jane Irwin. Soft error and energy consumption interactions: A data cache perspective. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 132–137, Aug 2004.
- [73] Xiaojun Lin and Ness B. Shroff. The impact of imperfect scheduling on cross-layer congestion control in wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 14(2):302–315, 2006.
- [74] Xiaojun Lin, Ness B. Shroff, and R. Srikant. A tutorial on cross-layer optimization in wireless networks. 24(8):1452–1463, Aug. 2006.
- [75] Jacob R. Lorch and Alan Jay Smith. Reducing processor power consumption by improving processor time management in a single-user operating system. In *Proceedings of the 2nd*

- annual international conference on Mobile computing and networking (MobiCom)*, pages 143–154, Nov. 1996.
- [76] Dominic Lucchetti, Steven K. Reinhardt, and Peter M. Chen. Extravirt: detecting and recovering from transient processor faults. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 1–8, New York, NY, USA, 2005. ACM Press.
- [77] Ritesh Mastipuram and Edwin C. Wee. *Soft Errors' Impact on System Reliability*. <http://www.edn.com/article/CA454636>, Sep 2004.
- [78] Jens Meggers, Gregor Bautz, and Anthony Sang-Bum Park. Providing video conferencing for the mobile user. In *IEEE Conference on Local Computer Networks*, page 526, March 1996.
- [79] Milly Watt. *Milly Watt Project*. Duke University, <http://www.cs.duke.edu/ari/millywatt/>.
- [80] Subhasish Mitra, Norbert Seifert, Ming Zhang, Quan Shi, and Kee Sup Kim. Robust system design with built-in soft-error resilience. *IEEE Computer*, 38(2):43–52, Feb 2005.
- [81] Kartik Mohanram and Nur A. Touba. Partial error masking to reduce soft error failure rate in logic circuits. In *DFT03*, pages 433–440, 2003.
- [82] S. Mohapatra, R. Cornea, H. Oh, K. Lee, M. Kim, N. Dutt, R. Gupta, A. Nicolau, S. Shukla, and N. Venkatasubramanian. A cross-layer approach for power-performance optimization in distributed mobile systems. In *Next Generation Software Program in conjunction with IPDPS*, page 218.1, April 2005.
- [83] Shivajit Mohapatra, Radu Cornea, Nikil Dutt, Alex Nicolau, and Nalini Venkatasubramanian. Integrated power management for video streaming to mobile handheld devices. In *Proceedings of the 11th annual ACM international conference on Multimedia*, pages 582–591, 2003.
- [84] Shivajit Mohapatra, Nikil Dutt, Alex Nicolau, and Nalini Venkatasubramanian. Dynamo: A cross-layer framework for end-to-end QoS and energy optimization in mobile handheld devices. 25(4):722–737, May 2007.

- [85] K. Mohr and L. Clark. Delay and area efficient first-level cache soft error detection and correction. In *IEEE International Conference on Computer Design (ICCD)*, 2006.
- [86] MPEG. <http://www.cselt.it/mpeg/>.
- [87] Shubhendu S. Mukherjee, Joel Emer, Trygve Fossum, and Steven K. Reinhardt. Cache scrubbing in microprocessors: Myth or necessity? In *PRDC'04*, 2004.
- [88] Shubhendu S. Mukherjee, Christopher Weaver, Joel Emer, Steven K. Reinhardt, and Todd Austin. A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In *MICRO*, Dec 2003.
- [89] O. Musseau. Single-event effects in soi technologies and devices. *IEEE Trans. on Nuclear Science*, Apr 1996.
- [90] Chris T. K. Ng, Deniz Gündüz, Andrea J. Goldsmith, and Elza Erkip. Minimum expected distortion in gaussian layered broadcast coding with successive refinement. In *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 2226–2230, June 2007.
- [91] M. Nicolaidis. Time redundancy based soft-error tolerance to rescue nanometer technologies. In *VTS'99*, 1999.
- [92] A. K. Nieuwland, S. Jasarevic, and G. Jerin. Combinational logic soft error analysis and protection. In *IOLTS06*, 2006.
- [93] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile application-aware adaptation for mobility. *SIGOPS Operating System Review*, 31(5):276–287, 1997.
- [94] NS2. Network Simulation version 2, <http://www.isi.edu/nsnam/ns/>.
- [95] Richard Phelan. Addressing soft errors in arm core-based designs. Technical report, ARM, 2003.
- [96] D. K. Pradhan. *Fault-Tolerant Computer System Design*. Prentice Hall, 1996. ISBN 0-1305-7887-8.

- [97] Qinru Qiu, Qing Wu, and Massoud Pedram. Dynamic power management in a mobile multimedia system with guaranteed quality-of-service. In *Proceedings of the 38th conference on Design automation (DAC)*, pages 834–839, 2001.
- [98] Nhon Quach. High availability and reliability in the Itanium processor. *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 61–69, Sep–Oct 2000.
- [99] Vijay T. Raisinghani and Sridhar Iyer. Cross-layer design optimization in wireless protocol stacks. *27:720–724*, Oct. 2003.
- [100] Srinivas Ramanathan, P. Venkat Rangan, and Harrick M. Vin. Frame-induced packet discarding: An efficient strategy for video networking. In *Network and Operating System Support for Digital Audio and Video*, pages 173–184, 1993.
- [101] George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, and David I. August. Swift: Software implemented fault tolerance. In *CGO '05: Proceedings of the international symposium on Code generation and optimization*, pages 243–254, Washington, DC, USA, 2005. IEEE Computer Society.
- [102] P. Roche, G. Gasiot, K. Forbes, Oapos, V. Sullivan, and V. Ferlet. Comparisons of soft error rate for SRAMs in commercial SOI and bulk below the 130-nm technology node. *IEEE Trans. on Nuclear Science*, 50(6), Dec 2003.
- [103] Mihaela Van Der Schaar and Sai Shankar N. Cross-layer wireless multimedia transmission: Challenges, principles, and new paradigms. *12(4):50–58*, Aug. 2005.
- [104] Sanjay Shakkottai, Theodore S. Rappaport, and Peter C. Karlsson. Cross-layer design for wireless networks. *41(10):74–80*, Oct. 2003.
- [105] Prashant Shenoy and Peter Radkov. Proxy-assisted power-friendly streaming to mobile devices. In *Proceedings of SPIE/ACM Multimedia Computing and Networking Conference (MMCN)*, 2003.
- [106] P. Shivakumar and N. Jouppi. CACTI 3.0: An Integrated Cache Timing, Power, and Area Model. In *WRL Technical Report 2001/2*, 2001.



- [107] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on soft error rate of combinational logic. In *DSN02*, 2002.
- [108] Aviral Shrivastava, Ilya Issenin, and Nikil Dutt. Compilation techniques for energy reduction in horizontally partitioned cache architectures. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, pages 90–96, 2005.
- [109] Vineet Srivastava and Mehul Motani. Cross-layer design: A survey and the road ahead. 27:112–119, Dec. 2005.
- [110] Blaine Stackhouse, Brian Cherkauer, Mike Gowan, Paul Gronowski, and Chris Lyles. A 65nm 2-billion-transistor quad-core Itanium processor. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 92–93, 598, 2008.
- [111] Mark Stemm and R. H. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Science, Special Issue on Mobile Computing*, 80(8):1125–1131, Aug. 1997.
- [112] Weilian Su and Tat L. Lim. Cross-layer design and optimization for wireless sensor networks. In *Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD)*, pages 278–284, 2006.
- [113] Synopsys Inc., Mountain View, CA, USA. *Design Compiler Reference Manual*, 2001.
- [114] Clark N. Taylor, Sujit Dey, and Debashis Panigrahi. Energy/latency/image quality tradeoffs in enabling mobile multimedia communication. In *Proc. of Software Radio: Technologies and Services*, pages 55–66. Springer Verlag, Jan 2001.
- [115] L. Tong, V. Naware, and P. Venkitasubramaniam. Signal processing in random access. *IEEE Signal Processing Magazine*, 21(5):29–39, Sep. 2004.
- [116] A. Tourapis, G. Shen, M. Liou, O. Au, and I. Ahmad. A new predictive diamond search algorithm for block based motion estimation. In *Proc. of Visual Comm. and Image Proc. (VCIP)*, June 2000.

- [117] Mihaela van der Schaar and Deepak S. Turaga. Cross-layer packetization and retransmission strategies for delay-sensitive wireless multimedia transmission. *IEEE Transactions on Multimedia*, 9(1):185–197, Jan. 2007.
- [118] Mehmet C. Vuran and Ian F. Akyildiz. Cross-layer analysis of error control in wireless sensor networks. In *IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON)*, pages 585–594, Sep 2006.
- [119] Shuai Wang, Jie Hu, and Sotirios G. Ziavras. On the characterization of data cache vulnerability in high-performance embedded microprocessors. In *SAMOS*, 2006.
- [120] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos. Review of error resilient coding techniques for real-time video communications. *IEEE Signal Processing Magazine*, 17(4):61–82, July 2000.
- [121] Yao Wang and Qin-Fan Zhu. Error control and concealment for video communication: A review. 86(5):974–997, May 1998.
- [122] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. Scheduling for reduced CPU energy. In *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation (OSDI)*, pages 13–23, Nov. 1994.
- [123] S. Worrall, A. Sadka, P. Sweeney, and A. Kondoz. Motion adaptive error resilient encoding for mpeg-4. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 1389–1392, May 2001.
- [124] F. Wrobel, J. M. Palau, M. C. Calvet, O. Bersillon, and H. Duarte. Simulation of nucleon-induced nuclear reactions in a simplified SRAM structure: Scaling effects on SEU and MBU cross sections. *IEEE Trans. on Nuclear Science*, 48(6):1946–1952, 2001.
- [125] xTune. *Online Verifiable Cross-Layer Adaptation for Distributed Real-Time Embedded Systems*. University of California at Irvine, <http://xtune.ics.uci.edu/>.
- [126] Jie Xu and Brian Randell. Roll-forward error recovery in embedded real-time systems. In *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, page 414, 1996.

- [127] Kai-Chao Yang, Chun Ming Huang, and Jia-Shung Wang. Error resilient GOP structures on video streaming. *Journal of Visual Communication and Image Representation*, 18, 2007.
- [128] Wanghong Yuan and Klara Nahrstedt. A middleware framework coordinating processor/power resource management for multimedia applications. In *Proceedings of IEEE Global Telecommunications Conference (GlobeCom)*, Nov. 2001.
- [129] Wanghong Yuan and Klara Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. 37(5):149–163, Dec 2003.
- [130] Wanghong Yuan and Klara Nahrstedt. Practical voltage scaling for mobile multimedia devices. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 924–931, 2004.
- [131] Wanghong Yuan, Klara Nahrstedt, Sarita V. Adve, Douglas L. Jones, and Robin H. Kravets. Design and evaluation of a cross-layer adaptation framework for mobile multimedia systems. In *Proceedings of SPIE/ACM Multimedia Computing and Networking Conference (MMCN)*, January 2003.
- [132] Wanghong Yuan, Klara Nahrstedt, Sarita V. Adve, Douglas L. Jones, and Robin H. Kravets. Grace-1: Cross-layer adaptation for multimedia quality and battery energy. *IEEE Transactions on Mobile Computing*, 5(7):799–815, 2006.
- [133] Heng Zeng, Carla S. Ellis, Alvin R. Lebeck, and Amin Vahdat. ECOSystem: managing energy as a first class operating system resource. *SIGPLAN Notices*, 37(10):123–132, 2002.
- [134] Rui Zhang, Shankar L. Regunathan, and Kenneth Rose. Video coding with optimal inter/intra-model switching for packet loss resilience. *IEEE Journal on Selected Areas in Communications*, 18(6):966–976, June 2000.
- [135] Wei Zhang. Computing cache vulnerability to transient errors and its implication. In *DFT'05*, 2005.
- [136] Wei Zhang. Replication Cache: A Small Fully Associative Cache to Improve Data Cache Reliability. *IEEE Computers*, 54(12):1547–1555, Dec 2005.

- [137] Wei Zhang, Sudhanva Gurumurthi, Mahmut Kandemir, and Anand Sivasubramaniam. ICR:In-Cache Replication for Enhancing Data Cache Reliability. In *DSN*, 2003.
- [138] Y. Zorian, V. A. Vardanian, K. Aleksanyan, and K. Amirkhanyan. Impact of soft error challenge on SoC design. In *IEEE International Symposium on On-Line Testing (IOLTS)*, pages 63–68, 2005.
- [139] M. Zorzi, J. Zeidler, A. Anderson, B. Rao, J. Proakis, L. A. Swindlehurst, M. Hensen, and S. Krishnamurthy. Cross-layer issues in MAC protocol design for MIMO Ad Hoc networks. *13(4):62–76*, Aug. 2006.