

Mitigating the Impact of Hardware Failures on Multimedia Applications – A Cross-Layer Approach

Kyoungwoo Lee¹, Aviral Shrivastava², Minyoung Kim¹, Nikil Dutt¹, Nalini Venkatasubramanian¹

¹Department of Computer Science
School of Information and Computer Sciences
University of California, Irvine, CA 92697, USA

{kyoungwl, minyoungk, dutt, nalini}@ics.uci.edu

²Department of Computer Science and Engineering
School of Computing and Informatics
Arizona State University, Tempe, AZ 85281, USA

Aviral.Shrivastava@asu.edu

ABSTRACT

Increasing exponentially with each technology generation, hardware-induced soft errors pose a significant threat for the reliability of mobile multimedia devices. Since traditional hardware error protection techniques incur significant power and performance overheads, this paper proposes a cross-layer cooperative approach that exploits existing error control schemes at the application layer to mitigate the impact of hardware defects. Specifically, we propose Error Detection Codes in hardware, Drop and Forward Recovery in middleware, and error-resilient video encoding at the application level to effectively and efficiently combat soft errors with minimal overheads. Experimental evaluation on standard test video streams demonstrates that our cooperative error-aware method for video encoding improves performance by 60% and energy consumption by 58% with even better reliability at the cost of only 3% quality degradation on average, as compared to an ECC-based hardware protection technique. Combining intelligent schemes to select a recovery mechanism can help guide system designers trade off multiple constraints such as performance, power, reliability, and QoS, and provides the ability to perform system level design space exploration across these interdependent constraints.

1. MOTIVATION

Increasing exponentially with each technology generation, soft errors will soon become an everyday concern [8, 29]. Soft errors are transient faults that are caused due to a variety of deep submicron reasons, including sudden voltage drops, signal interference, random noise, etc., but cosmic radiation strike causes more soft errors than all other reasons put together [2]. Soft errors are emerging as a problem in mobile multimedia devices, since these consumer devices – for competitive market reasons – increasingly deploy components manufactured using the latest technology, and operate at low voltages for extended battery life. Both of these factors reduce the threshold charge $Q_{critical}$ for a striking radiation particle to cause a soft error, which in turn greatly affects the reliability. Since memories occupy majority real estate on-chip, memories are most vulnerable to soft errors. Mobile multimedia devices are especially prone to soft errors owing to i) the sheer volume of data pro-

cessed in multimedia applications, and ii) they are more likely to be used in environments with high soft error rates, e.g., mountain tops and airplanes.

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% [15] and power consumption by up to 22% [20] 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 QoS¹), to the best of our knowledge, ours is the first attempt at a cross-layer approach for achieving reliability, and trading off reliability for power/QoS in mobile devices.

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 automated recovery of the detected errors, we deploy error recovery solutions in the middleware. Traditionally, on receiving an erroneous frame, the middleware will request retransmission 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 [27]. While BER can result in significant power and performance overhead, DFR can result in significant loss in QoS. Therefore in this paper, we propose and explore hybrid approaches of using DFR and BER to achieve low overheads in power and performance without much loss in QoS. Furthermore, we exploit an energy-efficient error-resilient technique at the application layer to improve the QoS (e.g., Probability-Based Power Aware Intra Refresh or PBPAIR [11]). Our cross-layer approach is also able to exploit specialized microarchitectural features for reliability (e.g., a Partially Protected Cache or PPC [13]) in a seamless manner. We show that this cross-layered approach is effective in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

¹For instance, GRACE [7, 32] deploys cross-layer methods for maximizing power reduction with the satisfactory QoS; DYNAMO [5, 6, 18, 19] proposes a proxy-based middleware approach for trading off video QoS and power; and xTune [10] performs cross-layer online timing-QoS verification at the proxy server.

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 corrections scheme.

The specific contributions and results of our work are:

- We propose a cooperative, error-aware approach that effectively exploits existing error control schemes across system layers. Our cooperative approach extends the applicability of existing error control schemes at the application abstraction layer to mitigate the impact of hardware defects at the hardware abstraction layer.
- To assist our cooperative approach, we present a middleware solution that triggers error control schemes with an appropriate error translation, and which selects an appropriate recovery policy (BER or DFR) based on information available in a mobile video encoding system.
- Our cooperative, cross-layer protection (CC-PROTECT) exploits an error-resilient video encoding (PBPAIR [11]) and a DFR mechanism with an EDC-based PPC [13] architecture that does not incur overheads in terms of power and performance. Rather, our proposal reduces the access latency of memory subsystem by 61%, the energy consumption of memory subsystem by 52%, the failure rate by about 1,000× at the cost of less than 1 dB of video quality, compared to a traditional video encoding running on a data cache without protection.

2. A CROSS-LAYER APPROACH TO SUPPORT RELIABILITY AND QOS

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 power, error-prone networks).

Techniques have been developed to enable QoS in multimedia applications executing in error-prone networks. At the application layer, error-resilient video encoding techniques enable adaptive encoding of information based on knowledge of network conditions [4, 11]. 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 represents 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 [21] 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) [13] 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 in the protected cache.

In this paper, our goal is to exploit the limited error tolerance of mobile multimedia applications to enhance their reliability to hardware-level "failures" without creating an adverse impact on power/performance profile at the device level or sacrificing application QoS. *We believe that addressing such power/performance/reliability/QoS tradeoffs in the presence of hardware failures 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 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 1 reveals 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.

2.2 Related Work

Existing work already demonstrates the effectiveness of cross-layer methods for mobile multimedia as opposed to schemes isolated at a sin-

Table 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 System Failure
Protection	Error-Resilience, Error-Concealment, etc.	Triple Modular Redundancy, Error Correction Codes, etc.
Error Metric	Packet Loss Rate (%)	Soft Error Rate (FIT^a)
Time Perspective	Dynamic	Transient

^a FIT (Failures In Time): the number of failures in 10⁹ operation hours

gle abstraction layer [5, 6, 7, 10, 18, 19, 32]. Yuan et al. [31] 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) [32] to coordinate adaptation of multimedia applications and CPU speeds for mobile multimedia systems. Mohapatra et al. [18] 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. [10] 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 reference model have been widely investigated as a promising optimization tools to efficiently reduce the energy consumption, especially transmission energy consumption, in wireless multimedia communications [1, 26, 25]. Vuran et al. [26] 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. [25] 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. Bajic [1] developed cross-layer error control schemes considering joint source rate selection and power management for wireless video multicast.

Our work 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 and their impacts. As illustrated earlier, this issue is a leading concern for embedded architectures of the future. Secondly, we will show how to exploit the cross-layer methodology to activate error control schemes at one abstraction layer to combat errors at a different abstraction layer.

2.3 The Cooperative Cross-Layer Approach

We conjecture that a dual pronged approach is needed to effectively address the aforementioned reliability/QoS/performance/power tradeoffs. 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 will focus on transient hardware errors (soft errors), i.e., those that do not *immediately* cause a permanent failure of the system². To create error-awareness, we consider the presence of inexpensive EDC 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 3.1. Hence our problem reduces to that of developing 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 DFR 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

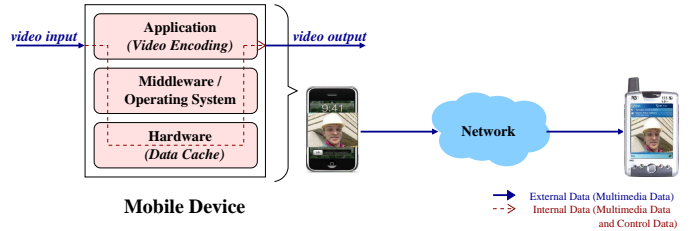


Figure 1: System Model - Mobile Video Encoding System

and a data cache as shown in Figure 1. Video encoding can be *error-prone* or *error-resilient*; a data cache can be designed to be *error-prone* or *error-protected*. Different compositions vary with respect to overall performance, power, QoS, and reliability. 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, 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 since it incurs high overheads due to expensive protection.

Given the ability to support error-awareness through less expensive EDC schemes, our strategy is to use the information on SER (soft error rates) to

1. Bypass potential failures by triggering error recovery mechanisms which reinitialize the erroneous data cache, and simultaneously
2. Reinforce application data using error-resilient encoding mechanisms by translating the SER metric into the input metric of the encoding algorithm (e.g. the network packet loss rate).

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

In particular, we explore a Drop-and-Forward Recovery (DFR) mechanism (shown in Figure 3(c)) that drops a current encoding frame and moves forward to the next frame encoding once an error is detected in a mobile video encoding system. The DFR mechanism works well with an EDC scheme to improve power and performance significantly while increasing reliability as well. As discussed, EDC is less expensive than ECC [16] and overheads due to checkpoints are negligible [30] while EDC can be as immune to soft errors as ECC with respect to reliability. In addition, dropping an erroneous frame potentially improves performance and energy reduction since it skips expensive processing algorithms for the frames.

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 of Backward Error Recovery (BER) mechanisms that rolls backward and re-encodes the current frame once an error is detected as shown in Figure 3(b).

3. CC-PROTECT - COOPERATIVE CROSS-LAYER STRATEGIES FOR FAILURE HANDLING

In this section, we present specific CC-PROTECT - a middleware driven approach for cooperative composition of cross-layer strategies to

²We also scope out the impact of software bugs introduced by programmers at the application/middleware/OS layers

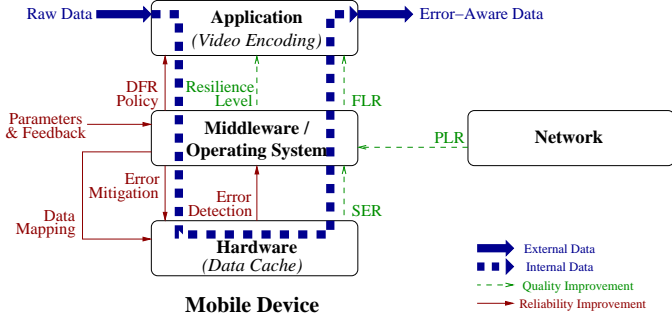


Figure 2: A Cross-Layer, Error-Aware Method: mitigating hardware defects with minimal costs by using error-resilience and a drop and forward recovery (DFR) in a video encoding

support error resilience. Figure 2 illustrates the 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 on a video encoding application. Soft error rates, obtained by error detection techniques at the hardware layer are communicated to the middleware which then

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

We instantiate two specific strategies for error recovery and error-resilience within CC-PROTECT. The specific error metric we use and evaluate in our study is soft error rate (SER). First, to mitigate the impacts of soft errors on the video quality, we exploit a power-aware error-resilient encoding technique, PBPAIR [11]. We present a simple, intuitive and effective translation of SER into Frame Loss Rates (FLR) used in turn by the error-resilient PBPAIR encoding strategy. Next, we exploit our prior work on partially protected caches to design a naive DFR mechanism for PPCs [13]. Using information captured in the middleware, we then extend the naive mechanism to achieve a balance between DFR and BER in Section 3.3.

3.1 Error-Resilient Video Encoding for QoS Improvement

Error-resilient video encoding techniques have been developed to reduce the impact of transmission errors, e.g., packet losses, on the video quality [4, 11, 28]. The PBPAIR (Probability-Based Power Aware Intra Refresh) technique [11] addresses the tradeoff between energy-efficiency and compression-efficiency based, 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 reduce compression efficiency by increasing the number of intra-coded macro-blocks in the video when packet loss rates are high. Inherently PBPAIR is energy-efficient (when errors are higher) and adaptive since its resilience can be adjusted for various PLRs. PBPAIR takes two parameters – *Packet Loss Rates (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. 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 a 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. 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 2.

3.2 Drop and Forward Recovery Mechanism for Reliability Improvement

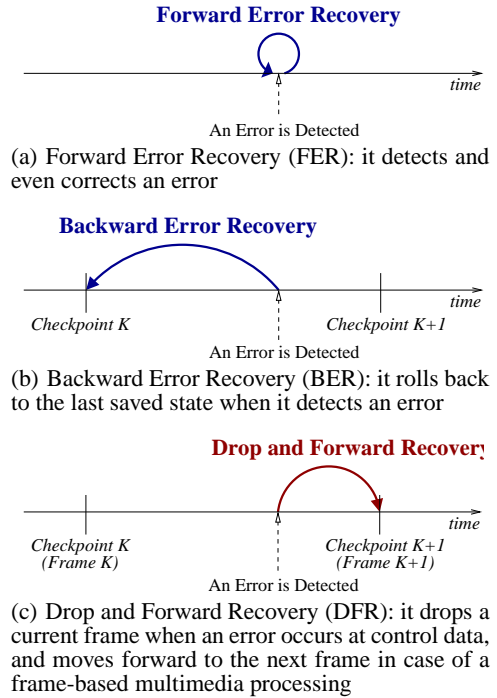


Figure 3: Error Recovery Mechanisms

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) [21] according to when an error is recovered as shown in Figure 3(a) and Figure 3(b). We explore the use of Drop and Forward Recovery or DFR (See Figure 3(c)) that combines error detection mechanisms with checkpoints to discontinue processing of the current frame and initiate processing of the next checkpointed frame.

Our objective is to apply DFR techniques on a cache architecture that uses PPCs (partially protected caches). A PPC architecture consists of two caches at the same level of memory hierarchy with 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 – the runtime problem is one of mapping information into the two respective caches. In our design, the protected cache is equipped with EDC that only detects errors and hence improves power and performance [16]. Since multimedia data itself does not cause a system failure [13], multimedia data is exposed to soft errors by being mapped into the unprotected cache in a PPC.

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 encoded frame ($K + 1^{th}$ frame) in Figure 3(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 [12]. We next discuss extensions to the naive DFR scheme to overcome quality losses when they occur.

3.3 Selective DFR Mechanisms

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 device.

Slack-Aware DFR/BER

The main problem with a BER approach is the subsequent loss of QoS due to violation of multimedia real-time guarantees. 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 quality improvement. Since encoding time is highly non-deterministic and varies from frame to frame, our scheme presents a knob to select a policy based on the elapsed time with ACET (Average Case Execution Time). 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 if the elapsed time from the start of the frame K encoding, $T_{elapsed} = T_{error} - T_K$, is smaller than given threshold time, $T_{threshold}$. 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 selecting the BER policy, 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

In this policy, we exploit our knowledge of frame characteristics and its impact on the video quality to determine whether to apply DFR or BER when a frame is erroneous. For example, I-frames are more critical than P-frames with respect to video quality [4, 11]. 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. Otherwise, it drops the current frame and moves forward to the next frame (DFR). Based on available information at the embedded system, 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.

The frame-aware policy also maintains and uses history, e.g., whether the previous frame has been dropped or not due to a soft error, in the recovery policy. If the previous frame was dropped, FA-DFR/BER prevents the current frame from being dropped since 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 the difference between two frames, higher the impact on video quality if the current frame is lost. Thus, if the difference between them, $diffFrames$, is larger than given threshold value, $diffThresh$, FA-DFR/BER selects BER. Otherwise, DFR is selected.

QoS-Aware DFR/BER

The potential problem with a DFR mechanism is the significant degradation of the QoS due to several frame drops. QA-DFR/BER (QoS-Aware DFR/BER) selects the BER policy when the obtained QoS does not meet the QoS requirement. The accumulated PSNR value, $QoS_{accumulated}$, 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 current frame if $QoS_{accumulated}$ is less than $QoS_{threshold}$, given threshold QoS value in PSNR. Otherwise, the default policy, DFR, is selected. Note that QoS here refers to the encoder, considering decoder QoS must incorporate knowledge of transmission errors and is beyond the scope of this paper.

4. EXPERIMENTAL SETUP

System Compositions To demonstrate the effectiveness of our cross-layer, cooperative scheme to combat soft errors, we develop 5 system compositions, shown in Table 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) encoding [4]. 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 just 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 is provided in hardware. This is implemented through the use of Error Correction Code (ECC) in Partially Protected Cache architecture [13]. As compared to protecting the whole cache, PPCs provide efficient reliability by just protecting the non-multimedia data against soft errors. This composition presents the low failure rate and high QoS, as it protects at hardware level. However, it incurs high overheads in terms of power and performance.
3. **APP-PROTECT:** In this composition, all error detection and correction is provided in the application. For this, we use error-resilient video encoding PBPAIR [11]. 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 in order to ensure a fair comparison.
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.
5. **CC-PROTECT:** This is our proposed composition, in which we use error-resilient video encoding, PBPAIR, and PPC with EDC scheme, and 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 when an error is detected.

Table 2: System Compositions - Our CC-PROTECT is a middleware-driven, cooperative approach aware of hardware failures

System Compositions with respect to Error Resilience					
Abstraction Layers	BASE	HW-PROTECT	APP-PROTECT	MULTI-PROTECT	CC-PROTECT
Application	GOP (error-prone encoding)	GOP (error-prone encoding)	PBPAIR (error-resilient encoding)	PBPAIR (error-resilient encoding)	PBPAIR (error-resilient encoding)
Middleware	None	None	◦Monitor network errors & Inform PBPAIR of PLR	◦Monitor network errors & Inform PBPAIR of PLR	◦Monitor network errors & Inform PBPAIR of PLR •Translate SER for PBPAIR •Trigger Selective DFR (Drive cache update & Inform PBPAIR)
Operating System	None	◦Map pages to a PPC	None	◦Map pages to a PPC	◦Map pages to a PPC •Monitor soft errors
Hardware	Unprotected Cache (error-prone cache)	PPC with ECC (error-protected cache)	Unprotected Cache (error-prone cache)	PPC with ECC (error-protected cache)	PPC with "EDC" (error-protected cache)

GOP:Group-Of-Picture,PBPAIR:Probability-Based Power Aware Intra Refresh,PLR:Packet Loss Rate,SER:Soft Error Rate,DFR:Drop and Forward Recovery,PPC:Partially Protected Cache, ECC:Error Correction Codes (*expensive in terms of power and performance, e.g., a Hamming Code (38,32)*),EDC:Error Detection Codes (*much less expensive than ECC, e.g., a parity code*)

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

FOREMAN₁.QCIF, . . . , and FOREMAN₇₄.QCIF. And we ran a simulation at least four times with each sequence, and thus more than 300 runs have been 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 Slack-Aware DFR/BER in Section 3.2.

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 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 2.28×10^{-17} at 90 nm^3) is much less than this accelerated SER by several orders of magnitude, but it increases exponentially as technology scales [2, 8, 17, 29]. 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.

4.1 Simulation Setup

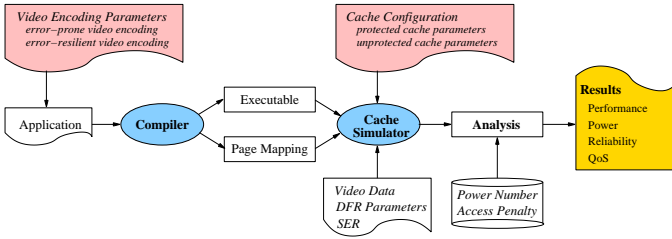


Figure 4: Experimental Setup – Compiler/Simulator/Analyzer

We perform our study on an extensive simulation environment that we have built to model the HP iPAQ h5555 [9] like processor-memory system. We have modified the *sim-cache* simulator from the SimpleScalar toolchain [3] to model the PPC architecture and to inject soft errors as in [13]. To support the unequal protection for a PPC architecture, compiler as shown in Figure 4 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 (176×144 pixels) are used for our simulation study, and each of them represents a video clip of low activity, medium activity, and high activity. 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). For example, 300 frames of *FOREMAN.QCIF* are separate into *FOREMAN₀.QCIF*,

4.2 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 memory subsystem as shown in Figure 4. QoS is measured in PSNR with the encoded video output and the original video input.

Performance Model: For performance comparison of each composition, we estimate the access latency to the memory subsystem using the statistics generated by the cache simulator as shown in Figure 4. 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 [22] and the Synopsys Design

³It is projected using the increasing ratio of 1,000 FIT/Mbit at 180 nm technology and 100,000 FIT/Mbit at 130 nm technology [2, 8, 17].

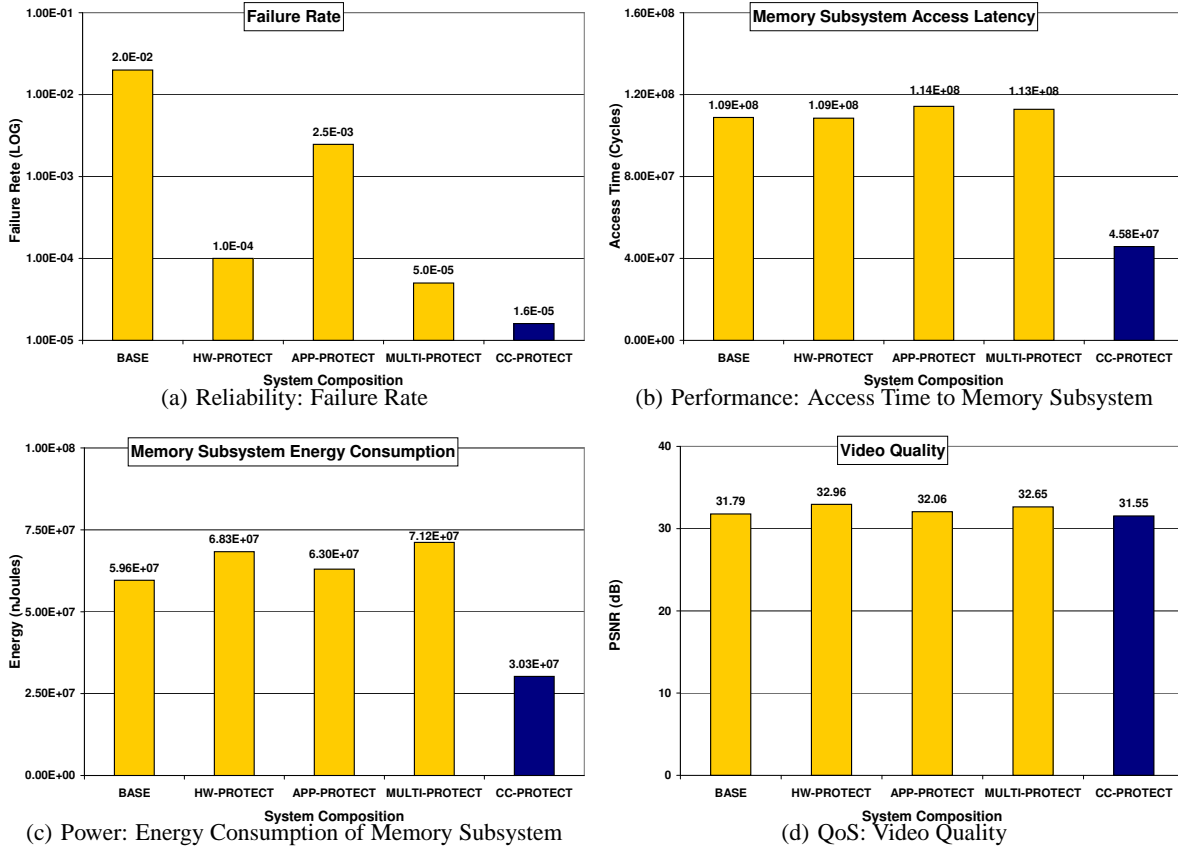


Figure 5: **CC-PROTECT** achieves the low-cost reliability at the minimal QoS degradation

Compiler [24] as in [13], and the overhead of delay for EDC is calculated using the ratio between delays of ECC and EDC from [13, 16]. 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 in Figure 4.

Energy Model: We estimate the energy consumption of the memory subsystem using the power models presented in [23]. The overheads of power for a Hamming code (38,32) and a parity code are synthesized and estimated similar to those of delay. The power consumption penalty for a recovery policy such as DFR and BER is estimated through the simulations. The energy consumption of the memory subsystem E 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.

Failure Rate Model: To estimate reliability, we define an execution a *Success* if it ends within twice of a normal execution time and returns the correct output opened by a decoder. Otherwise, it is a *Failure* 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 following binomial distribution analysis.

Quality of Service Model: We estimate the QoS in PSNR (Peak Signal to Noise Ratio). PSNR is defined in dB as $PSNR = 10 \log_{10} \left(\frac{MAX^2}{MSE} \right)$, 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 the correctly reconstructed output (without errors).

5. EXPERIMENTAL RESULTS




We present two sets of experiments. First, we demonstrate the effectiveness of our cross-layer, error-aware methods in low-cost reliability at the slight cost of QoS for different video streams (Section 5.1). Second, we show the effectiveness of intelligent DFR/BER selection schemes to improve the video quality (Section 5.2).

5.1 Effectiveness of CC-PROTECT

Figure 5 clearly shows that our cross-layer, error-aware approach increases the reliability with the minimal costs of performance and energy consumption at the minimal degradation of video quality.

Figure 5(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 mainly because 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 them. This is because it has less time to be exposed to soft errors due to a frame drop and the performance efficiency of PBPAIR than GOP. It is important that APP-PROTECT (composed of PBPAIR and an unprotected data cache) shows the close failure rate 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 composi-

Table 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
		CC-PROTECT	0.27	0.34	0.1 E-2	1.02
<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
		CC-PROTECT	0.42	0.51	0.1 E-2	0.99
<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
		CC-PROTECT	0.49	0.58	0.1 E-2	0.93

tions.

Figure 5(b) shows that our CC-PROTECT 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 our 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 the performance overhead but CC-PROTECT improves the performance. This performance improvement is because 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 as it shows 5% overhead compared to BASE because PBPAIR increases the compression efficiency rather than the performance efficiency at low PLR such as 0% PLR. With 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 [13].

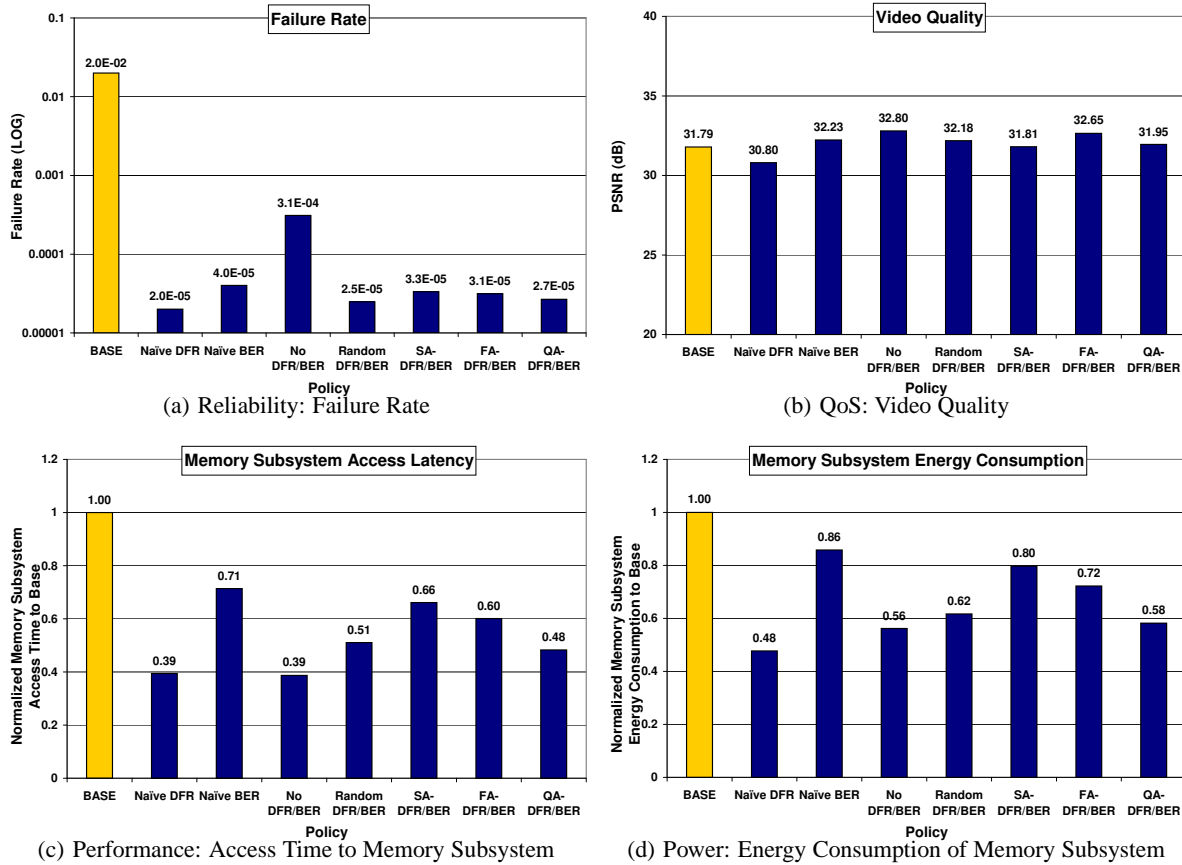
With the perspective of energy consumption of memory subsystem, our CC-PROTECT saves the energy consumption by 49%, 56%, 52%, and 57% as compared to BASE, HW-PROTECT, APP-PROTECT, and MULTI-PROTECT, respectively, as shown in Figure 5(c). Cross-Layer approach reduces the energy consumption of memory subsystem because of (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-MBs (Macroblock) than expensive inter-MBs. Note that all other compositions incur overheads of performance as well as power compared to BASE except for CC-PROTECT. Thus, our CC-PROTECT can even reduce the power and access time of memory subsystem while obtaining the high reliability.

Our CC-PROTECT achieves video quality close to those of other compositions as shown in Figure 5(d). While an EDC scheme protects the non-multimedia data in our 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 that these results come from only one soft error at the protected cache in a PPC (tens errors in the unprotected cache), and the video quality may degrade significantly due to multiple frame drops resulting from multiple occurrences of soft errors. We will present the experimental results in those cases in Section 5.2.

Table 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*, (ii) the minimal protection onto control data at a protected cache with an EDC, and (iii) 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 the similar results under the various network status. 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-MBs, causing high performance in PBPAIR algorithms), as compared to BASE. Also, we have 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 [14].

In summary, our cooperative, error-aware methods exploit a DFR mech-



Selective Scheme	BASE	Naive DFR	Naive BER	No DFR/BER	Random DFR/BER	SA-DFR/BER	FA-DFR/BER	QA-DFR/BER	
Video Encoding	GOP	PBPAIR							
Recovery Policy (threshold value)	None	Always DFR	Always BER	None	Randomly DFR or BER (50%)	DFR or BER (Slack > 60%)	DFR or BER (Frame $\neq 2^{n^d}$)	DFR or BER (QoS > 31.79 dB)	
Data Cache	Unprotected Cache	PPC with EDC		PPC (No Protection)	PPC with EDC				

Figure 6: Intelligent selective schemes maintain the video quality and reliability with minimal overheads of power and performance

anism 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% on average.

5.2 Effectiveness of Intelligent Selective Schemes

Our CC-PROTECT 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 6 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 6, 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. And we parameterize a policy selection based on available information in a mobile embedded system. Naive DFR scheme shows the worst video quality as shown in Figure 6(b) at the least costs in terms of power and performance as shown in Figure 6(c) and Figure 6(d). Note that Naive DFR in Figure 6 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(d). On the other hand, Naive BER scheme presents the better video quality than that of 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 6(a). This is mainly because Naive BER increases the execution time, causing the 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 6(a). Note that Figure 6(b) 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 the 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 6 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^{nd} , 3^{rd} , and 4^{th} frame in the descending order of the importance on average. Thus, FA-DFR/BER with 2^{nd} 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^{nd} 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.

6. DISCUSSION

Reliability is of paramount concern in mobile embedded systems where the resources such as power and performance are limited. 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 way.

Traditionally reliability techniques have been developed at individual levels, and have remained seemingly incognizant of the strategies employed at other levels. While focussing 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. By synergistic cooperation between EDC at the hardware level, DFR and BER at the middleware, and PBPAIR at the application level on mobile devices, we obtain high reliability, high performance, and high energy saving at the cost of slight QoS degradation.

In addition to the experiments presented in this paper, we also explored combination of the proposed techniques, i.e., Slack, Frame, and QoS Aware DFR/BER (SFQA-DFR/BER). Combined approaches with this small set of threshold values expand the design space to improve the performance by up to 29% further at the cost of less than 2 dB QoS degradation. In summary, our cross-layer methods aware of error control schemes and errors can guide system designers to explore the interesting tradeoff spaces in terms of power, performance, reliability, and QoS for resource-constrained mobile devices, which were not discovered by previous approaches.

Our future work includes the extended cross-layer approach considering end-to-end devices in distributed real-time systems, and varying error control schemes with different error models across system abstraction layers. Also, intelligent design space algorithms will be investigated to efficiently guide system designers for exploiting our cross-layer schemes.

7. REFERENCES

- [1] I. V. Bajic. Efficient cross-layer error control for wireless video multicast. 53(1):276–285, Mar 2007.
- [2] R. Baumann. Soft errors in advanced computer systems. *IEEE Design and Test of Computers*, pages 258–266, 2005.
- [3] D. Burger and T. M. Austin. The SimpleScalar Tool Set, version 2.0. *SIGARCH Computer Architecture News*, 25(3):13–25, 1997.
- [4] L. Cheng and M. E. Zarki. PGOP: An error resilient techniques for low bit rate and low latency video communications. In *Picture Coding Symposium (PCS)*, Dec 2004.
- [5] DYNAMO. *Power Aware Middleware for Distributed Mobile Computing*. University of California at Irvine, <http://dynamo.ics.uci.edu/>.
- [6] FORGE Project. *A Framework for Optimization of Distributed Embedded Systems Software*. University of California at Irvine, <http://www.ics.uci.edu/forge/>.
- [7] GRACE Project. *Global Resource Adaptation through CoopErAtion*. University of Illinois at Urbana-Champaign, <http://rsim.cs.uiuc.edu/grace/>.
- [8] 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.
- [9] Hewlett Packard, <http://www.hp.com>. *HP iPAQ h5555 Series - System Specifications*.
- [10] 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.
- [11] 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.
- [12] K. Lee, M. Kim, N. Dutt, and N. Venkatasubramanian. Error exploiting video encoder to extend energy/QoS tradeoffs for mobile embedded systems. In *DIPES*, 2008.
- [13] K. Lee, A. Shrivastava, I. Issenin, N. Dutt, and N. Venkatasubramanian. Mitigating soft error failures for multimedia applications by selective data protection. In *CASES*, Oct 2006.
- [14] K. Lee, A. Shrivastava, M. Kim, N. Dutt, and N. Venkatasubramanian. Cross-layer interactions of error control schemes in mobile multimedia systems. Technical report, University of California at Irvine, Apr 2008.
- [15] J.-F. Li and Y.-J. 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.
- [16] L. Li, V. Degalahal, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. Soft error and energy consumption interactions: A data cache perspective. In *ISLPED*, Aug 2004.
- [17] R. Mastipuram and E. C. Wee. *Soft Errors' Impact on System Reliability*. <http://www.edn.com/article/CA454636>, Sep 2004.
- [18] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N. Venkatasubramanian. Integrated power management for video streaming to mobile handheld devices. In *ACM international conference on Multimedia*, 2003.
- [19] 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.
- [20] R. Phelan. Addressing soft errors in arm core-based designs. Technical report, ARM, 2003.
- [21] D. K. Pradhan. *Fault-Tolerant Computer System Design*. Prentice Hall, 1996. ISBN 0-1305-7887-8.
- [22] P. Shivakumar and N. Jouppi. CACTI 3.0: An Integrated Cache Timing, Power, and Area Model. In *WRL Technical Report 2001/2*, 2001.
- [23] A. Shrivastava, I. Issenin, and N. Dutt. Compilation techniques for energy reduction in horizontally partitioned cache architectures. In *CASES*, pages 90–96, 2005.
- [24] Synopsys Inc., Mountain View, CA, USA. *Design Compiler Reference Manual*, 2001.
- [25] M. van der Schaar and D. S. Turaga. Cross-layer packetization and retransmission strategies for delay-sensitive wireless multimedia transmission. *IEEE Transactions on Multimedia*, 9(1):185–197, Jan. 2007.
- [26] M. C. Vuran and I. 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.
- [27] Y. Wang and Q.-F. Zhu. Error control and concealment for video communication: A review. 86(5):974–997, May 1998.
- [28] S. Worrall, A. Sadka, P. Sweeney, and A. Kondoz. Motion adaptive error resilient encoding for MPEG-4. In *ICASSP*, May 2001.
- [29] 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), 2001.
- [30] J. Xu and B. Randell. Roll-forward error recovery in embedded real-time systems. In *ICPADS*, page 414, 1996.
- [31] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. 37(5):149–163, Dec 2003.
- [32] W. Yuan and K. Nahrstedt. Practical voltage scaling for mobile multimedia devices. In *ACM international conference on Multimedia*, pages 924–931, 2004.