CODING OVER RESOURCE-CONSTRAIED WIRELESS NETWORKS

By

Rami Halloush

A DISSERTATION

Submitted to
Michigan State University
In Partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Electrical Engineering

2012

ABSTRACT

CODING OVER RESOURCE-CONSTRAINED WIRELESS NETWORKS

By

Rami Halloush

In practice, wireless networks operate under multiple, mostly severe, constraints (bandwidth, energy resources, etc). Consequently, efficient techniques have to be employed in communicating data with sufficiently high data rates (depending on the application) while complying with the imposed constraints.

Distributed Video Coding (DVC) and Network Coding (NC) are amongst the most dominant techniques employed in constrained data networks. In this dissertation, we address these two techniques with the objective of realizing practical and efficient data networking solutions that fit in resource constrained wireless networks. In one part of the dissertation we address DVC over Visual Sensor Networks (VSNs) from a practical point of view, i.e., unlike a large body of research work related to this topic where the focus is on theoretical analysis and simulation, we study the practical aspects that arise when deploying DVC over real visual sensors. To that end, we develop a Resource-constrained DVC (RDVC) codec, deploy it over some of the widely used visual sensors, and conduct precise energy measurements that are used throughout our study.

One RDVC-related challenge that we address in this dissertation is source rate estimation, i.e., trying to efficiently identify the source rate to be used in encoding a frame. This question is crucial since sending more bits than necessary will lead to inefficiency in compression while sending fewer bits will lead to failure in decoding. We

propose a practical solution that completely eliminates the need for the costly feedback messages.

Another challenge we address is the global choice between a DVC encoding option that involves intensive computations and leads to less transmission versus another choice with minimal computations that implies higher transmission-energy overhead. We carry out an operational energy-distortion analysis for a variety of options available to RDVC on visual sensors.

Polar codes are the first codes proven to achieve capacity while having low encoding and decoding complexity. This motivates us to employ polar codes in our DVC platform. We compare the performance of polar codes with the more established and more investigated Low Density Parity Check Accumulate (LDPCA) codes in the context of DVC. Our results show that polar codes offer a clear advantage when coding smaller size image blocks. Consequently, polar codes could represent a viable solution for distributed sensor networks that capture low-resolution video signals.

In the final part of the dissertation, we survey state-of-the-art NC solutions designed to achieve high throughput transmission in multicasting over wireless mesh networks while maintaining 100% packet delivery ratio. We propose HopCaster; a scheme that employs the cache-and-forward transport strategy. We show that HopCaster achieves significant performance gains compared to schemes that employ the end-to-end transport strategy.

ACKNOWLEDGMENTS

I would like to show my gratitude to my advisor Dr. Hayder Radha for his valuable guidance.

I also would like to thank my Ph.D committee members; Dr. Subir Biswas, Dr. Selin Aviyente, and Dr. Sandeep Kulkarni for their helpful advices and feedbacks.

I am extremely grateful to my parents, brothers and sisters for their continuous support and patience. Special thanks to my brother (and former colleague at WAVES lab) Mohammed for the generous help he provided me during my study.

I would like to thank my friends for the great time I spent with them at MSU.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Dissertation Outline

In Chapter 1 we provide a brief background about Distributed Source Coding (DSC) and Network Coding (NC). We briefly overview the theoretical bounds of DSC, and we explain the syndrome-based DSC scheme that we employ in our study. For NC we provide an example that demonstrates how NC can be used to achieve reliability while maintaining high throughput.

In Chapter 2 we describe the design and implementation of a Resource-constrained distributed video coding (RDVC) codec that we deployed over the popular Micaz/Cyclops and Imote2/IMB400 visual platforms. Further, we report a summary of accurate measurements regarding the distribution of energy consumption over the different DVC operations. We carry out an operational energy-distortion analysis for a variety of options available to RDVC on visual sensors. This aspect addresses the global choice between intensive computations that lead to less transmission versus minimal computations that implies higher transmission-energy overhead.

In Chapter 3 we present our approach towards a rate-adaptive RDVC solution that eliminates the need for feedback messages from the base-station to the visual sensors; this solution is crucial for any realistic mapping of DVC over visual sensors.

In Chapter 4 we employ and evaluate the performance of polar codes in DVC. Further, we compare the performance of polar codes with the more established and more investigated Low Density Parity Check Accumulate (LDPCA) codes in the context of DVC.

In Chapter 5 we propose HopCaster, a novel protocol that incorporates intra-flow NC with hop-by-hop transport to achieve high-throughput reliable multicast over wireless mesh networks. We show that by adopting the hop-by-hop transport strategy, HopCaster avoids many problems incurred by the traditional end-to-end transport strategy.

In Chapter 6 we provide a conclusion and outline directions for future work.

Chapter 1

Introduction

Since Distributed Video Coding (DVC) is based on the theory of Distributed Source Coding (DSC), we dedicate this chapter to providing a brief overview of DSC. Further, we provide a brief demonstration of using Network Coding (NC) to achieve reliability while maintaining high throughput over wireless networks.

1.1. Distributed Source Coding

Distributed Source Coding (DSC) is a problem in information theory. It deals with compressing two correlated sources *independently*; i.e., each source has no access to the other source (this explains the name *distributed*). Slepian and Wolf [1] provided the theoretical bounds for the DSC problem in 1973. Let X and Y be two correlated independent and identically distributed (i.i.d) sources, then X and Y can be compressed at rates R_X and R_Y respectively with the following constraints:

$$R_X \geq H(X \mid Y)$$
$$R_Y \geq H(Y \mid X)$$
$$R_X + R_Y \geq H(X,Y)$$

Where $H(.|.)$ and $H(.,.)$ are the conditional and joint entropies respectively. Note that the aggregate rate can achieve the joint entropy, which is also achieved using *dependent* encoding where sources have access to each other (a practical example of joint encoding is compression using motion estimation and compensation). This means that by having independent encoding, DSC provides a compression scheme that has low computational complexity at the encoder side while theoretically achieving the same performance as the more complex joint encoding. Figure 1.1 illustrates the independent encoding of DSC versus the dependent joint encoding scenario.

The equations above define a set of achievable rates. These rates belong to the rate region shown in Figure 1.2. There are two cases for DSC; *symmetric* and *asymmetric*. In the symmetric case, sources are compressed using the same rate. This case is represented by the line emanating from the green circle in the figure. In the asymmetric case on the other hand, different rates are used to compress the sources. An important special case of the asymmetric case on which we focus in this dissertation is *compression with decoder side information*. In the latter case, one of the sources, say Y is compressed at rate bounded by the entropy $H(Y)$ and the other source is compressed at rate bounded by $H(X \mid Y)$. This case is represented by the two red circles in the figure (two corners of the rate region).

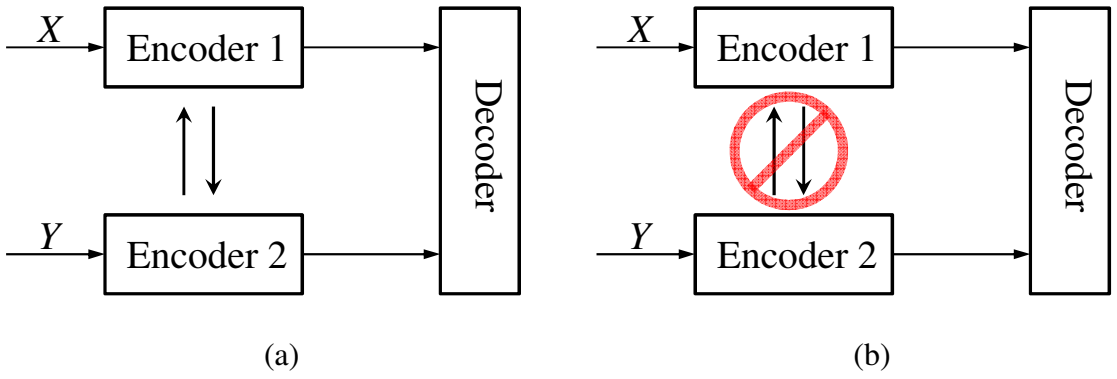


Figure 1.1: (a) Dependent encoding of two sources where each source has access to the other. (b) DSC's independent encoding where each source is completely isolated from the other. This reduces the encoding complexity (**For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation**).

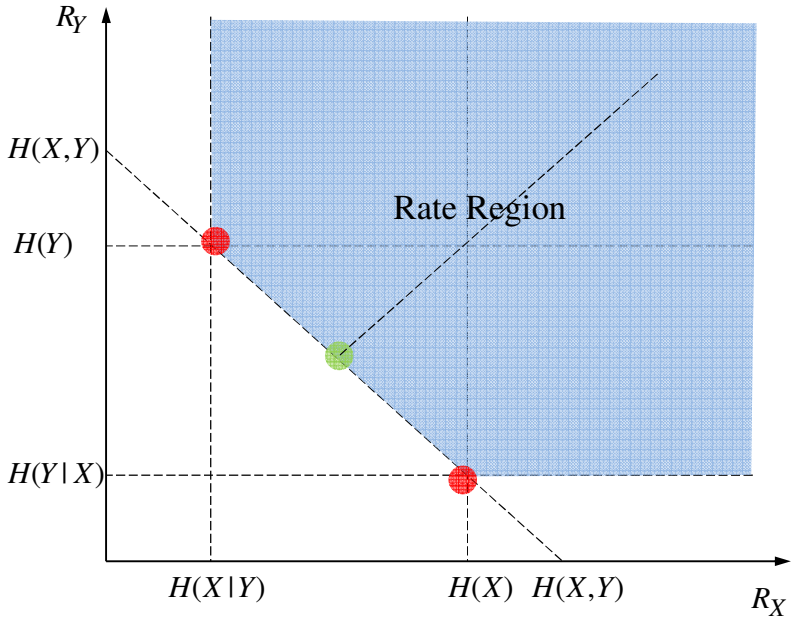


Figure 1.2: Rate region defined by the DSC constraints.

Syndrome-based distributed source coding [2] is a DSC paradigm that involves channel codes. Channel codes are involved in DSC since the disparity between the two sources to be compressed, X and Y, can be viewed as noise. In other words, X can be viewed as a noisy version of Y. Consequently, if Y is already communicated to the decoder, then the encoder can communicate X by transmitting a number of parity bits

sufficient to correct the noise in Y. Compression is achieved by transmitting the parity bits. In the next section we provide an example that explains the idea of syndrome-based DSC.

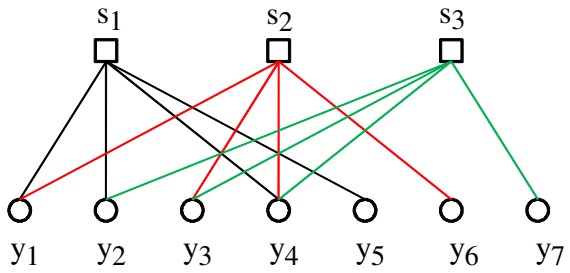
1.1.1. Syndrome-based Distributed Source Coding

Consider the following example where, at the encoder, we have two bitstreams to compress:

Y: 1 0 1 1 0 1 0 $\qquad\qquad$ X: 1 0 0 1 0 1 0

Note that the two bitstreams differ only in the third bit. Hence Y can be viewed as noisy version of X. If Y is already communicated to the decoder, then all what the encoder needs to do to communicate X is to correct the noise in Y. To do so, the encoder uses a channel code that maps X into a number of parity bits (or syndrome bits) that is sufficient to correct this noise in Y to get X back. Compression is achieved by sending parity bits rather than bits of X. For simplicity, we will consider the binary (7,4) Hamming code that is characterized by the following parity check matrix H and the corresponding Tanner graph [3]:

$$H = \begin{bmatrix} 1101100 \\ 1011010 \\ 0111001 \end{bmatrix}$$



To encode X, its bits are placed on the variable nodes of the graph (circle shaped nodes). At each check node (square shaped nodes), the XOR of the bits at the variable

nodes connected to that check node is computed. This results in the syndrome bit sequence s = 011 that is transmitted to the decoder (note that this is equivalent to multiplying X by H; $s^T = H \times X^T$). At the decoder, bits of Y are placed on the variable nodes of the Graph, and the received syndrome bits are fed into check nodes. At each check node, the XOR of the bits at the variable nodes connected to that check node is computed and compared to the syndrome bit fed to that check node. Bits of Y are flipped iteratively until the computed bits at check nodes match the syndrome bits (this iterative process is achieved through the Belief Propagation (BP) algorithm [3]). In our example, at the decoder, the computed bits at check nodes would be 000. BP will flip the third bit of Y and stop, because at that point bits at check nodes would be 011 which match the received syndrome bits, and X would be restored.

1.2. Network Coding

Network Coding (NC) [4] is a packet forwarding technique that aims to improve the performance of the traditional *store and forward* scheme where forwarding nodes merely propagate the packets they receive with no processing. In NC the role of the forwarding nodes is extended to include combining received/buffered packets before transmission. The end receiver node deduces the packets that were originally transmitted. NC aims to achieve the *maximum folw* [4]; which is the maximum data rate for a transmission between two nodes. According to the Max Flow Min Cut theorem by Fulkerson [5], for a multicast topology, the maximum flow equals the minimum cut. In [6] it is proved that linear coding is sufficient to achieve the upper bound. In linear network coding, forwarding nodes generate coded packets that consist of linear combinations of

previously received packets. The coefficients are chosen from a finite field and they are included in the coded packet prior sending in order to be used in decoding.

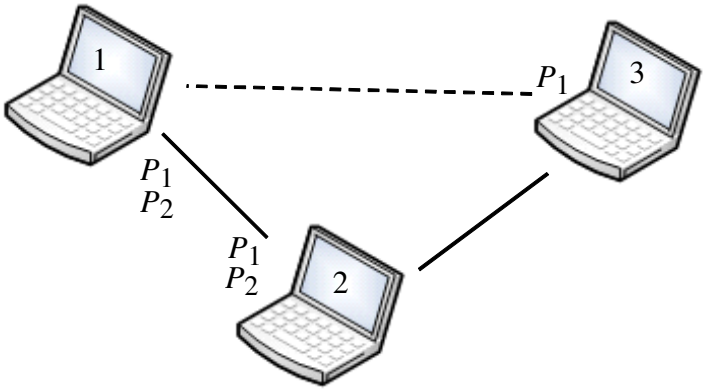


Figure 1.3: NC scenario to achieve reliability with high throughput.

In this dissertation we focus on using NC to achieve reliability in multicast protocols over wireless networks. The advantage of NC in this context is made clear by examining the example illustrated in Figure 1.3. The figure shows a simple multicast scenario where node 1 is a source node, node 3 is a destination node, and node 2 is an intermediate forwarding node. In this example, node 1 needs to send two packets, P_1 and P_2, to node 3. When broadcasting the two packets, node 2 manages to receive both of them, whereas node 3 overhears packet P_1. If node 2 is aware that packet P_1 is available at node 3, then it will forward packet P_2 only, and this will be the optimal behavior. Otherwise, it will send out both P_1 and P_2, which will incur one redundant transmission (P_1). Using NC eliminates the need to keep track of packets at node 3. Node 2 simply sends a linear combination of P_1 and P_2, e.g., $P=P_1+P_2$. Upon receiving P, node 3 will be able to construct P_2 by simply subtracting P_1 from P. In this example, reliability is achieved; as

100 % packet delivery ration (PDR) is satisfied, in the meanwhile redundant transmissions are eliminated resulting in high throughput.

1.3. Dissertation Contribution

The following is a summary of the contributions of this dissertation:

- We describe the design and implementation of a resource-constrained distributed video coding (RDVC) codec that we deployed over real visual sensor platforms, namely, the popular Micaz/Cyclops and Imote2/IMB400 visual platforms. Further, we report a summary of accurate measurements regarding the distribution of energy consumption over the end-to-end chain of RDVC within an actual sensor, including frame-capture, Discrete Cosine Transform (DCT), syndrome computation, and bit-stream transmission. The codec and energy measurements are used to highlight practical aspects related to DVC when deployed over real visual sensors.
- We present a practical approach towards a rate-adaptive RDVC that enables the source node to efficiently identify the source rate required to compress a frame at a certain point of time. The presented solution completely eliminates the need for feedback messages from the base-station to visual sensors; this solution is crucial for any realistic mapping of DVC over visual sensors.
- We carry out an operational energy-distortion analysis for a variety of options available to RDVC on visual sensors; this aspect addresses the global choice between intensive computations that lead to less transmission versus minimal computations that implies higher transmission-energy overhead.

- The recently discovered Polar codes [7] have many interesting features, e.g., they are the first codes proven to achieve capacity. Further, they have low encoding and decoding complexity. As polar codes are relatively recent, there has been little work to assess the potential of polar codes for practical applications. We employ polar codes in Distributed Video Coding (DVC). We compare the performance of polar codes with the more established and more investigated Low Density Parity Check Accumulate (LDPCA) codes in the context of DVC.
- We propose HopCaster; a Network Coding (NC) scheme for wireless networks. NC schemes that employ the conventional end-to-end transport strategy suffer degradation in their performance. The reason is that by employing the end-to-end transport strategy, forwarding nodes are unable to accurately determine the number of coded packets that they need to forward, and a large number of redundant packets will be injected into the network. HopCaster employs a novel hop-by-hop reliable multicast transport scheme that exploits in-network caching. The need for estimating the number of coded packets at forwarding node is completely eliminated, which results in performance gains. In addition, HopCaster is integrated with a cross-layer rate adaptation scheme that optimizes the multicast throughput performance based on varying receiver topology at different transmission stages.

Chapter 2

Distributed Video Coding Over Real Visual Sensors

Despite the vast body of work covering the theoretical, analytical, and simulation-based aspects of Distributed Video Coding (DVC), there has been little work focusing on the practical aspects of DVC over actual sensors. In this chapter, we provide a rather comprehensive description of key aspects of a Resource-constrained DVC (RDVC) framework that stems from our experience in mapping DVC solutions on actual visual sensors. In particular, (1) we describe the design and implementation of a RDVC codec that we deployed over the popular Micaz/Cyclops and Imote2/IMB400 visual platforms. (2) We report a summary of accurate measurements regarding the distribution of energy consumption over the end-to-end chain of RDVC within an actual sensor, including frame-capture, DCT and syndrome computation, and bit-stream transmission. (3) We carry out an operational energy-distortion analysis for a variety of options available to RDVC on visual sensors; this aspect addresses the global choice between intensive

computations that lead to less transmission versus minimal computations that implies higher transmission-energy overhead.

2.1. Introduction

Motivated by advances in low power image sensing technologies, and wireless sensor networking, a new paradigm of ubiquitous monitoring has emerged: Visual Sensor Networks (VSN). Typically, a VSN consists of resource constrained nodes (motes) and a more powerful node referred to as a base station. Motes periodically sense their surrounding and propagate their sensed visual content to the base station. Motes are usually battery operated; hence to achieve an acceptable mote lifetime the overall power consumption needs to be minimized. There are two main sources of power consumption in a mote: (a) computational processing and (b) data transmission. To reduce the transmission power consumption, visual sensor data has to be compressed without requiring significant computations and/or exchange of data among sensors. Therefore, for VSNs, a new compression paradigm, which employs low complexity *independent encoding* at the encoder side and delegates intensive computations to the decoder side (the powerful base station), has emerged. This paradigm is called Distributed Video Coding (DVC) [2] [8] and is based on the theory of Distributed Source Coding (DSC) pioneered by Slepian and Wolf [1] and Wyner and Ziv [9].

The importance of DVC as a compression paradigm that fits well in VSNs attracted researchers and led to significant advances (e.g., [2] [8]). A variety of extensions to early DVC approaches have also been proposed (e.g., [10] [11] [12]). Many research efforts addressing DVC employ theoretical and analytical formulations, while others use

simulations to derive results and conclusions. Interestingly, there has been little work highlighting the practical aspects of DVC over real visual sensor platforms and devices (e.g., [13] [14]). This fact motivated us to conduct a study towards a Practical DVC (RDVC) platform deployed over actual visual sensors [15] [16]. As part of our study, we implemented a flexible syndrome-based [2] DVC codec and deploy it over different real visual sensor platforms, namely, the Micaz/Cyclops [17] [18] and the Imote2/IMB400 [18] platforms. This chapter is intended to share our experience in designing, implementing and deploying practical DVC over real visual sensor platforms. We use our implementation to outline key practical aspects that arise when a RDVC platform is targeted. One practical aspect that we present in this chapter is an operational energy-distortion analysis for a variety of options available to RDVC when it is deployed over an actual visual sensor; this aspect addresses the global choice between intensive computations that lead to less transmission versus minimal computations that implies higher transmission-energy overhead. Another important practical feature desired (arguably required) in any viable RDVC solution is the ability of the DVC encoder to identify the source rate of a Wyner-Ziv (WZ) frame without the need for receiving feedback messages from the decoder. A practical solution for this feature is described in the next chapter of this dissertation. It is apparent that studying the aforementioned practical aspects necessitates having an accurate power consumption measure for radio transmission and the different computational operations involved in DVC. Consequently, we present the approaches that we followed in measuring power, and later on how we employ our measurements in studying the different practical aspects of DVC. The remainder of this chapter is organized as follows. In Section 2.2, we describe the visual

sensor platforms that we use in our study. In Section 2.3 we describe our syndrome-based RDVC codec implementation. In Section 2.4 we outline the approaches we followed in evaluating power consumption of the codec when deployed over a visual sensor. In Section 2.5 we highlight the importance of having an operational power-distortion description for the different options available for a DVC codec in order to make a global choice amongst such options. In Section 2.6 we conclude the chapter and introduce next chapter.

2.2. Visual Sensor Platform

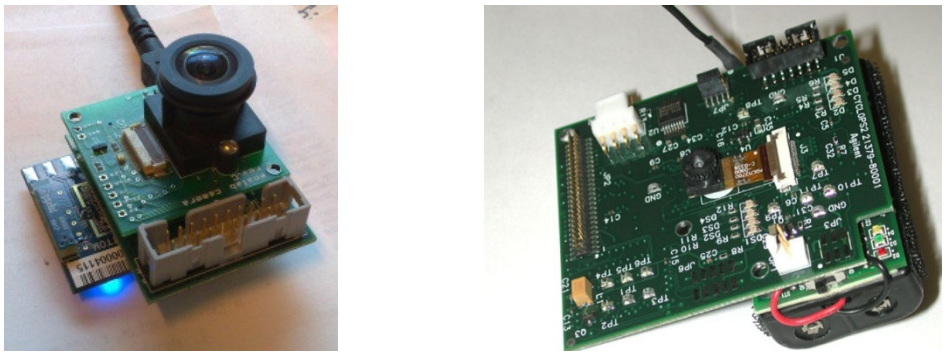
Many visual sensor platforms can be found in the market/literature. They basically differ in many aspects such as power consumption, image quality, physical dimensions, etc. We can categorize visual sensor platforms into two categories:

1. Sensor platforms that were originally designed for sensing scalar physical phenomena (such as temperature, humidity, etc.). In general, scalar applications have lower computational demands than visual applications; hence, scalar motes are limited in their resources. For instance, Micaz it is equipped with the 8-bit ATmega128L [19] microcontroller with clock speed set to 8MHz. It has 128K Bytes program flash memory, and 512K Bytes flash memory to store measurements. For radio communication, It uses the low power CC2420 RF transceiver [20], 2.4 GHz IEEE 802.15.4/Zigbee compliant [21]. To bridge the gap between the demanding visual application and the resource constrained mote, the image sensor is endowed with its own computational unit, and the underlying mote is merely responsible for data communication. For instance, Cyclops is equipped with an ATmega128L

microcontroller operating at 7.3728 MHz, and a complex programmable logic device (CPLD). For imaging, it is equipped with a CMOS Agilent ADCM-1700 CIF resolution imager in addition to 64 KB external SRAM memory to store captured images. Other visual senor platforms that belong to this category are the CMUcam3 [22], eCAM [23], and WiCa [24].

2. The second category includes sensor platforms that were designed to support multimedia applications. Motes in this category have extended capabilities in terms of computational units and memory size. Imote2 [18] is one example of such motes. Imote2 is equipped with the PXA271 XScale microprocessor [25] whose frequency can be scaled from 13 MHz up to 416 MHz. PXA271 includes a wireless MMX (WMMX) coprocessor to accelerate computations. Moreover, PXA271 includes 256 KB SRAM, 32 MB SDRAM, and 32 MB flash memory. Further, Imote2 interfaces with the IMB400 multimedia board [18] to form a visual sensor node. IMB400 is equipped with the OV7670 imager that can capture 640×480 resolution images. IMB400 only captures frames, and all processing is done on the underlying Imote2 node. Another mote that belongs to this category is Stargate [18]. For a comprehensive survey on sensor platforms, we refer the reader to [26].

In this chapter, we describe our experience in implementing and deploying RDVC over two different visual sensor platforms; Micaz/Cyclops, and the more recent Imote2/IMB400. Both Micaz and Imote2 are programmed using nesC [27], and they both run TinyOS [28] that is becoming a de-facto standard operating system (OS) for embedded sensor networks.



(a) (b)

Figure 2.1: sensor platforms employed in our study: (a) Imote2/IMB400 and (b) Micaz/Cyclops.

Figure 2.1 shows both the Imote2/IMB400 and the Micaz/Cyclops platforms that are used in our study.

2.3. Syndrome-Based DVC Codec over Real Visual Sensors

There has been a vast body of work related to DVC in the literature. PRISM [12] (Power-efficient, Robust, hIgh-compression, Syndrome-based Multimedia coding) is one of the earliest schemes. To encode a frame, it is first divided into bocks (16×16 or 8×8) over which Discrete Cosine Transform (DCT) is applied. Correlation noise between the block to be encoded and the co-located one in the previous frame is computed (the square error difference between DCT coefficients in the two blocks). The amount of correlation noise is used to classify the block being encoded into three categories: (1) Skip mode, when the amount of correlation noise is negligibly small. In this mode the block is not encoded at all. (2) Intra mode, when there is a significant amount of correlation noise. In this mode the block is intracoded. (3) DSC mode, when the correlation noise is relatively small and can be corrected through channel coding. In this case, the codeword space is

divided into cosets, each of which is labeled by a coset leader (syndrome). Compression is achieved by transmitting the syndrome, whose size depends on the correlation channel condition. In [12], a trellis channel code is used. To aid the decoder to perform motion estimation, the encoder generates a cyclic redundancy check (CRC) of the encoded block. The decoder uses different blocks residing in the frame memory as candidate side information for decoding. Each decoding attempt is followed by a CRC computation, and the decoding attempt whose CRC matches the received CRC is the one to be adopted.

Girod et al [29] [30] [31] [32] have contributed in developing a series of DVC schemes. In al [29], even and odd numbered frames are separated into two different groups. Pixel domain WZ encoding is applied on even numbered frames, whereas odd numbered frames are intracoded. Pixels are first quantized using a uniform scalar quantizer. Quantized pixels are then fed into a rate compatible punctured turbo (RCPT) code. Resulting parity bits are stored in a buffer. A feedback channel is maintained between the encoder and the decoder. The encoder sends parity bits in an incremental way in response to requests received from the decoder until the successful decoding is acknowledged. Many techniques are used to generate SI frames. The simplest is to find the average of the predecessor and successor of the WZ frame. More sophisticated techniques involve motion estimation. After the decoder decodes the WZ frame, it uses a minimum mean square error estimator (MMSE) to restore pixel values. In [30] the scheme is developed into a more general framework where SI frames are generated by interpolating or extrapolating frames residing in the decoder's frame memory (already decoded WZ frames or previous key frames). In [31] the framework is further extended to work in the transform domain where DCT is used. In [32], channel coding (turbo codes) is

applied on bitplanes of quantized DCT coefficients. Further, motion estimation (ME) at the decoder side is used to generate SI. To achieve that, the encoder generates helper motion estimation information. In particular, a hash codeword is generated at the encoder side and sent to the decoder. The hash codeword is simply a coarsely quantized and subsampled version of the block to be encoded. Similar to PRISM, the distance between the hash of the current block and the co-located one in the previous frame is calculated. If the distance exceeds a certain threshold, the hash codeword is sent. The decoder uses the hash to conduct a motion search that results in the best SI block.

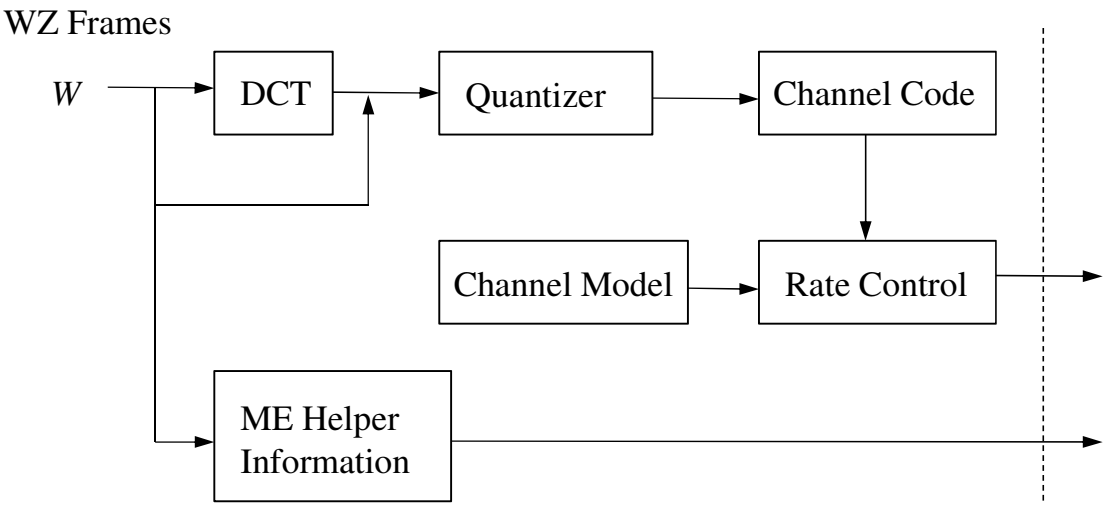


Figure 2.2: Block Diagram of an abstract WZ encoder.

From the description of the aforementioned schemes, we can identify the components of a WZ encoder. Figure 2.2 shows a block diagram of an abstract WZ encoder. To cope with the stringent nature of motes, the different modules shown in the figure have to be implemented in a way that ensures low complexity. We address this requirement by making the following observations:

1. In general, bitwise logical operations (such as AND, OR, and SHIFT) are considered to be inexpensive in terms of power consumption [33] compared to other operations (multiplication for instance). This reflects on the channel code module shown in Figure 2.2, as we design our codec to operate on the bit level (bitplanes of images). Here, a syndrome-based [2] Low-Density Parity-Check Code (LDPC) DVC [3] arises as an attractive solution (see Appendix A: Syndrome-Based DVC), where the core operation is a bitwise XOR between a bitplane and rows of a parity check matrix (as opposed to other DVC schemes that uses more complex operations such as modular arithmetic [34] [12]). Furthermore, our RDVC codec avoids the expensive floating point operations, and this reflects on the DCT module by employing the DCT described in

[35] where computations involve only integers, and expensive multiplications are replaced with shift operations.

2. The scarcity of memory is another reason that justifies employing a syndrome-based LDPC code in the channel code module, as the sparsity of the LDPC parity check matrix allows for compact representation and storage.
3. Different compression schemes in DVC have different underlying energy demands and, in return, provide different video quality scores. For example, the DCT component can be skipped, and compression is applied directly on raw pixels. Moreover, both DCT coefficients and pixels could be represented via Gray codes or regular binary codes. Our study shows that various combinations of these coding options provide better performance over different ranges of the energy-quality space [15]. For example, under certain conditions, the combination of raw-pixel and Gray coding can outperform other coding options (including DCT-based) as we show later in this chapter (Section 2.5).
4. For the rate control module, employing a feedback channel between the encoder and decoder may not be desirable in setups where DVC is deployed over real visual sensor networks, due to power considerations. In the next chapter, we present a practical solution to this problem over real visual sensors. Our solution results in using source rates close to the ideal rates while completely avoiding incremental transmissions and feedback messages.

The following is a brief description of our RDVC codec. We consider a setting where a frame captured by the encoder is classified into two categories:

a) Key (side information) frame: Captured at time instance n. We refer to a key frame by Y.

b) WZ frame: A frame that follows a key frame, i.e., it is captured at time instance n+1. We refer to a WZ frame by X. Therefore, our codec encodes captured frames in pairs. In each pair, the frame that was captured earlier is a key frame, and the following frame is a WZ frame.

2.3.1. Key Frame Intracoding

Key frames encoding can be achieved in many low-complexity ways following traditional still image codecs used in popular standards such the ones used in different JPEG and MPEG specifications. In summary, intracoding a Key frame Y starts with applying DCT to the frame. The resulting DCT coefficients are then quantized. The quantization process turns many DCT coefficients (the ones with small values) into zeros. Run Length Encoding (RLE) is then used to encode quantized coefficients. Finally, Huffman encoding is used to encode the output of RLE.

2.3.2. Wyner-Ziv Coding

Let X_i and Y_i denote the i^{th} bitplanes of WZ frame X and key frame Y respectively, where i ranges from 1 (least significant bitplane) to 8 (most significant bitplane). The following is a description of the encoding process: Key frame Y is intra-coded and communicated to the decoder. WZ frame X is DCT transformed (this can be skipped when DVC is applied on raw pixels). A uniform quantizer of $M = 2^b$ intervals is used to quantize

DCT coefficients (or pixels). Quantized values are then represented via Gray codes (or regular binary codes). The resulting frame is decomposed into b bitplanes. Next step is to use Low-Density Parity-Check Accumulate (LDPCA) [36] codes to encode the bitplanes (see APPENDIX A: LDPCA Codes). For all values of i, we scan the $h \times l$ size bitplanes X_i and Y_i into one dimensional vectors x_i and y_i with size $n \times 1$, where $n = h \times l$. x_i is then multiplied by the $n \times n$ parity-check matrix H:

$$S_i = H \times x_i \tag{2.1}$$

The $n \times 1$ vector S_i is referred to as the syndrome vector. Bits in S_i are then accumulated to generate a new $n \times 1$ vector that is denoted here as A_i [36]. A subset of bits is chosen from A_i to be sent to the decoder, where the number of transmitted bits depends on the disparity between X_i and Y_i. The larger the disparity the more bits the encoder needs to send (higher source rate[1]) and the smaller compression ratio is achieved. Identifying the appropriate source rate at the encoder is a crucial practical aspect that we will discuss in detail in the next chapter. For now, let's assume that the encoder transmits a sufficient number of bits $\overline{t_i}$ to the decoder. At the decoder side, bit accumulation is inversed to restore $\overline{t_i}$ syndrome bits. Further, a Tanner [3] graph that has $\overline{t_i}$ check nodes and n variable nodes is constructed for decoding x_i. Bits of y_i (representing the i^{th} bitplane of Y) are fed into the n variable nodes, whereas the $\overline{t_i}$ syndrome bits are fed into the check nodes, and belief propagation [3] algorithm runs on the graph to restore x_i. When this

[1] Here, the source rate is the number of parity bits needed for coding a WZ frame divided by the total number of bits in the WZ frame.

process is applied to all bitplanes, i.e., for all i from 1 to b, the decoded b bitplanes are combined together in one frame. The final step is performing inverse Gray coding and inverse DCT (if they are used) to approximate the encoded WZ frame.

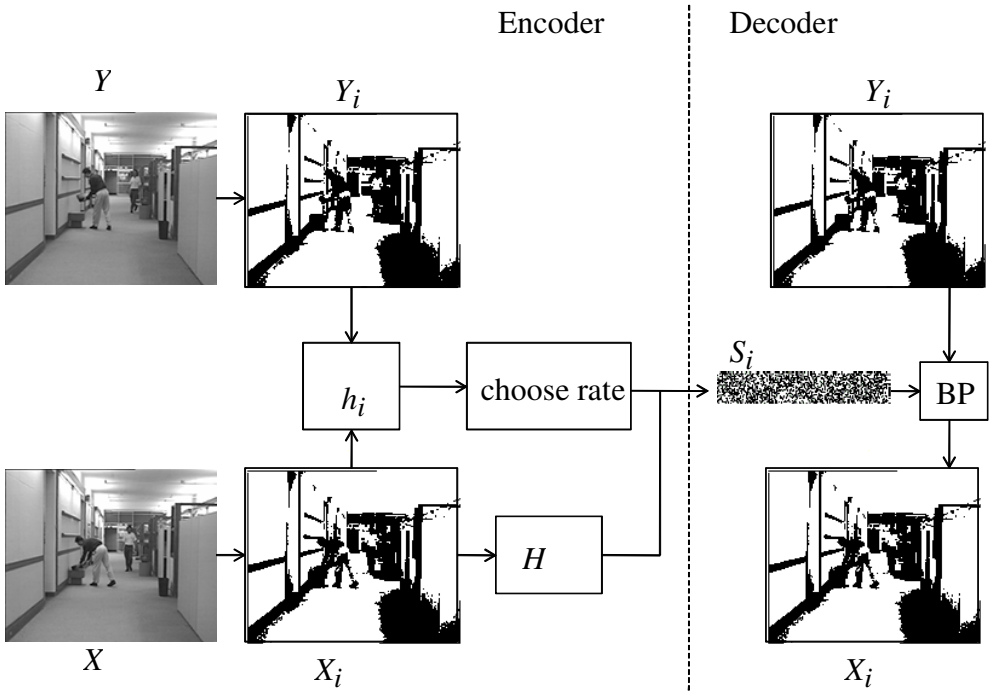


Figure 2.3: Pictorial demonstration of the DVC codec.

Figure 2.3 provides a pictorial illustration of the whole process. The role of h_i in the figure will become clear in the next chapter.

2.4. Energy Measurement Setup

Because motes are battery operated, their energy resource is quite limited. A practical DVC solution aims to utilize energy in an efficient manner in order to achieve an acceptable mote life time. This utilization takes the form of compression (and hence less

transmission) that has low computational complexity. Consequently, measuring power consumption of transmission and computation is crucial for studying practical DVC solutions. Our key objective here has been to evaluate the energy consumption of transmission as well as the different DVC computational operations in isolation of each other. Therefore, we have experimented with the following approaches:

1. A shunt resistor is placed in series between the visual sensor mote (Imote2/IMB400 node) and the power supply. Voltage across the shunt resistor is captured using an infiniium oscilloscope [37]. To capture small voltage levels, we use an instrumentation amplifier [38]. We account for the offset introduced by the amplifier in later energy calculations. To measure the energy consumption of a particular operation (transmission or computation), we need to mark the beginning and end of that operation. Therefore, we make utility of the general purpose pins available at the Imote2's microprocessor; we program our codec so that it asserts one pin (turns the signal at that pin on) at the beginning of the operation of interest, and deactivates that pin when the operation is over. By monitoring this *indicator signal* on one channel of the oscilloscope and the drawn current signal on another channel, we are able to accurately quantify the energy consumption of any operation. Power is computed as follows:

$$E = \sum_{i=1}^{T} P^i \qquad 2.2$$

   Figure 2.4 shows the current drawn by the Imote2/IMB400 node during compressing and transmitting a pair of a key and a following WZ frames. It also shows the time support of the different operations in the codec (attained via the indicator signal). Note that the current starts from its lowest level during capturing a frame, and then it

goes up during DCT and intracoding. Radio is turned off during those operations, and it is turned back on only upon transmission. Therefore, when intracoding is over, radio turns on, and transmission begins. When transmission ends, radio goes back off. For a WZ frame, the current starts from the lowest level during capturing the frame, then it rises during DCT and WZ encoding (denoted by DVC in the figure), and finally radio is turned on to transmit the syndrome bit stream. Table 2.1 summarizes the per frame energy cost for the different computational operations, in addition to the packet transmission cost. Note that transmission could take on a long period of time if data is not sufficiently compressed, and as energy is a function of time duration, DVC aims to reduce the volume of transmitted data, and hence reducing the time duration of transmission in order to achieve energy savings.

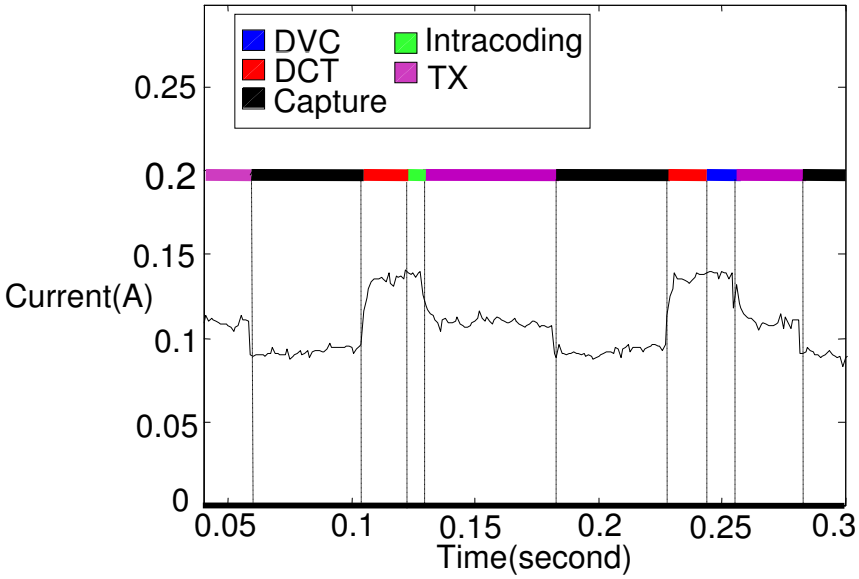


Figure 2.4: Current drawn during different operations (**For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation**).

Operation	Energy	Time Period
DCT	8.52 (mJ/frame)	15 (ms/frame)
DVC	5.53 (mJ/frame)	10 (ms/frame)
Capture	16.87 (mJ/frame)	44.25 (ms/frame)
Intracoding	3.27 (mJ/frame)	5 (ms/frame)
TX	3.4 (mJ/packet)	7.8125 (ms/ packet)

Table 2.1: Energy cost and time support of different operations.

2. As power consumption affects the mote's battery voltage level, a natural way to quantify power consumption would be through monitoring the battery's voltage level during the course of compression and transmission. Consequently, in [15] our approach is based on voltage discharge curves [39]. We modify the codec (deployed over the Micaz/Cyclops node) so that it executes only the operation of interest (DCT for instance) repeatedly. Meanwhile, the voltage level of the battery is sampled and recorded. This process continues until the voltage level reaches a certain threshold, where the encoder stops and the average cost per frame of the operation is computed. Figure 2.5 shows the voltage level decay over time for frame capturing, DCT, and DVC (syndrome generation). Table 2.2 summarizes the frame processing cost (of the different encoding operations) in addition to the packet transmission cost. By achieving sufficient compression, DVC aims to reduce the number of transmitted packets, and hence reducing energy consumption.

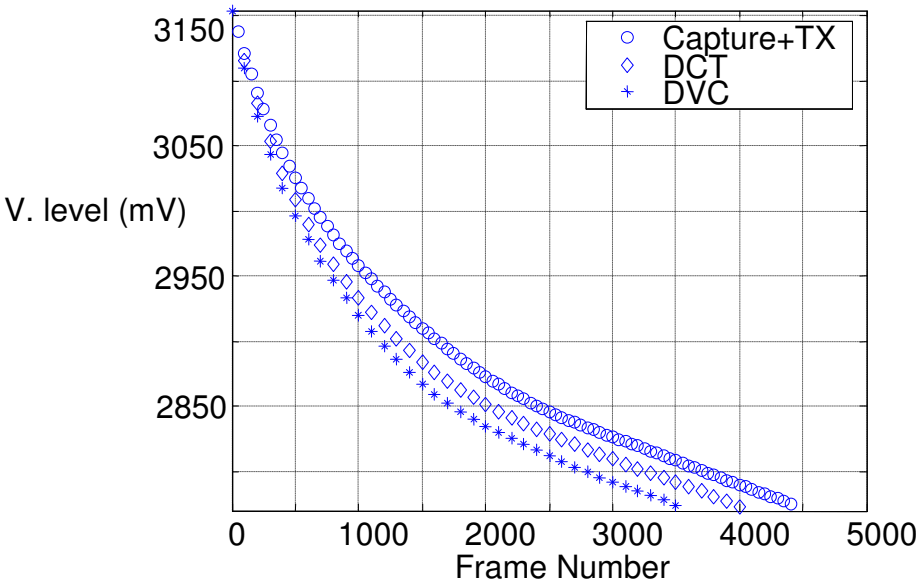


Figure 2.5: Battery voltage decay over time during different operations.

Operation	Voltage Decrease
Camera	8.87 (μV/frame)
DCT	8.81 (μV/frame)
DVC	21.27 (μV/frame)

TX	0.509 (μV/packet)

Table 2.2: Cost of transmission and computation operations.

2.5. Transmission-Computation Tradeoff

There are many options available for a RDVC codec when it is deployed over an actual visual sensor. Those options have different underlying energy requirements (computation and transmission energy), and they result in different video quality (distortion) scores. While a computational intensive scheme is costly in terms of computational power consumption, it achieves significant compression and hence transmission power saving. On the other hand, a less complex encoding scheme is less costly in terms of computational power consumption and more costly in terms of transmission power. Interestingly, as we show in [15], although the more complex scheme achieves more compression, and hence more transmission power saving, it may not be more efficient in terms of the overall power consumption (computation and transmission) beyond a particular achieved level of video quality. We show this by conducting an operational energy-distortion analysis involving two DVC codecs: (1) a DVC codec that incorporates DCT (extra computational power is spent on performing DCT), (2) and a DVC codec that runs directly on raw pixels (less power consuming than the DCT case, but less compression is achieved and hence more transmission power consumption). We compare between the two codecs when both are deployed over the

Micaz/Cyclops platform using measurements reported in Section 2.4. Both codecs are used to encode a video sequence that consists of 100 frames captured by the Cyclops camera. Frames have 64×64 resolution. Figure 2.6 compares between the two codecs. Each point in the figure corresponds to the number of bytes that resulted from encoding the 50 Wyner-Ziv frames using both the DCT and pixel encoding schemes along with different quantization levels with and without Gray codes. Note from the figure that when Gray coding is not used, DCT results in more data compression compared to the pixel scheme. This is also valid when Gray coding is used.

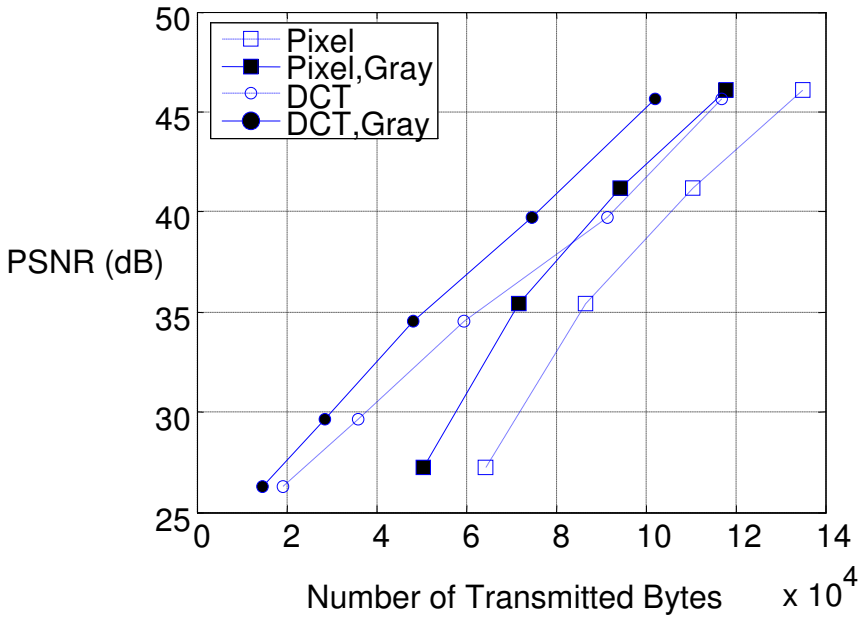


Figure 2.6: Compression performance of different encoding schemes.

As we are more interested in comparing between the two schemes in terms of the overall cost (compression and computation) at different video quality levels, we use the measurements presented in section 4 (for Micaz/Cyclops) to map points in Figure 2.6 to those in Figure 2.7. The overall cost at each point has three components: (1) The cost of capturing and applying DVC (computing syndromes) to the 50 Wyner-Ziv frames; this

component is common between DCT and pixel based schemes. (2) The cost of transmitting the encoded frames; this is different across the two schemes and it depends on the compression performance of each. (3) The cost of applying DCT; this is applicable only in the DCT case.

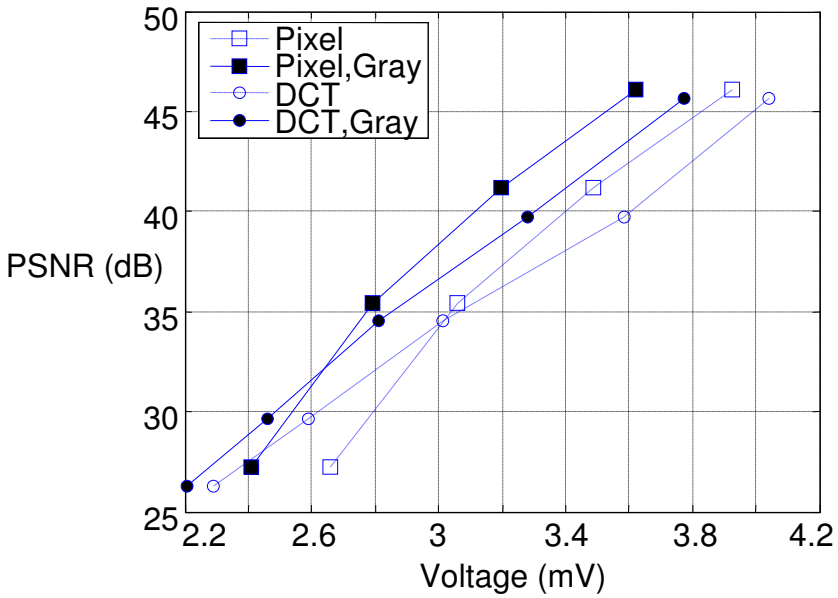


Figure 2.7: Power-distortion performance of different encoding schemes.

Figure 2.7 shows that for the case of no Gray coding, and for lower video quality values (below 34 dB), DCT demonstrates lower overall power consumption. In this case DCT achieves significant saving in transmission power such that the overall power (transmission, and computation) is still less than that in the pixel case. On the other hand, for PSNR values greater than 34 dB, the pixel based scheme demonstrates lower overall power consumption. In this case the transmission power saving of DCT is not significant enough to keep the overall power consumption less than that of the pixel case. The same justification applies for the case of Gray coding. Figure 2.7 provides an operational energy-distortion analysis for both encoding schemes. For any RDVC codec, it is extremely important to have this type of analysis when there is a set of options available

for the codec, and those options have different computational requirements, and they result in different quality scores. Having a power-distortion view of the different options assists in making a global optimal choice that accounts for tolerable video quality and power budget.

2.6. Discussion

In this chapter we described a RDVC codec that we deployed over the popular Micaz/Cyclops and Imote2/IMB400 visual platforms. Further, we used different ways to evaluate the power consumption of the codec over both platforms. We used the codec along with the power measurements to highlight some practical aspects of DVC over actual platforms and devices. In particular, we showed that when there is a variety of options available to RDVC, some are computationally intensive and lead to less transmission and others are less intensive and result in higher transmission-energy overhead. In this case having a power-distortion profile of the different options assists in making a global decision amongst them. In the next chapter, we use the RDVC codec and energy measurements described in this chapter to address another practical aspect; the problem of identifying the appropriate source rate at the encoder side without the need to exchange feedback messages with the decoder.

Chapter 3

Distributed Video Coding Based on Source Rate Estimation

In Distributed Video Coding (DVC) the encoder compresses frames at source rates that depend on the statistical dependency between the Wyner-Ziv (WZ) and side information frames. An important issue that we address in this chapter is providing the encoder with a mechanism to identify the source rate to be used in encoding a WZ frame (Here, the source rate is the number of parity bits needed/used for coding a WZ frame divided by the total number of bits in the WZ frame). One possible solution is to follow a feedback approach; the encoder starts by sending a small amount of data (low source rate). In case the decoder fails to recover the compressed frame, it provides the encoder with a feedback requesting more bits. This solution results in using source rates that are ideal in the sense that they are the minimal rates that lead to successful decoding. Nevertheless, this solution might not be practical for visual sensor networks as it may exhaust limited bandwidth and energy resources. In this chapter re present a mechanism to estimate ideal source rates without the need to exchanging feedback messages. The

proposed mechanism uses conditional entropy (entropy of a WZ source conditioned on a side information source) to estimate ideal source rates. Further, we show that by estimating the source rates (using optimal estimators) we can achieve a video quality and compression performance that is close to that achieved when feedback messages are exchanged. Moreover, we show that by avoiding incremental transmission and feedback messages, the proposed estimation-based approach can demonstrate lower energy consumption for a range of video quality compared with the feedback approach when both are deployed over a real visual sensor platform, namely, the imote2/IMB400.

3.1. Introduction

One important question is: How can the encoder know the number of parity bits that have to be sent to the decoder? Or, what is the source rate that it has to use in encoding a WZ frame? One possible solution is to use feedback messages between the decoder (base station) and the encoder (sensor node). Under the feedback approach, when coding a WZ frame, the encoder transmits the compressed data in an incremental fashion; it starts with a low source rate (small number of parity bits). If the decoder fails to restore the WZ frame using this number of bits, it (the decoder) sends the encoder a feedback message requesting more bits. Thus, the encoder incrementally increases the source rate until the decoder acknowledges successful decoding. This approach is considered to be *ideal* in the sense that it will eventually lead to successful decoding for all encoded WZ frames while requiring the minimal number of parity bits. Nevertheless this approach may not be desirable (or even feasible) in setups where DVC is deployed over real visual sensor networks, because limited communication bandwidth and power considerations may not

afford feedback messages and incremental transmission overhead. In this chapter, we present a practical solution for this problem over visual sensor networks. Our solution results in using source rates that are close to the ideal rates of the feedback approach while completely avoiding incremental transmissions and feedback messages. The approach we are proposing is to use the conditional entropy $H(X \mid Y)$ [40] to estimate ideal source rates. Conditional entropy gives information about the disparity between *X*, the WZ frame, and *Y*, the side information frame. As the source rate depends on this disparity, we expect the source rate to be statistically dependent on the conditional entropy, and hence, the encoder can use the conditional entropy to estimate the necessary source rate. In this chapter, we examine this dependency. Further, we build a DVC codec that employs two estimators: a Maximum a Posteriori (MAP) estimator, and a Minimum Mean Squared Error (MMSE) estimator. We use estimated source rates to encode different video sequences, and we show that the performance of the proposed estimation-based approach (in terms of compression and video quality) is close to that achieved when feedback messages are exchanged. Moreover, we show that due to the overhead incurred by employing the feedback approach in a real visual sensor application, the estimation-based approach could result in lower energy consumption for a range of video quality levels. To demonstrate that, we use the imote2/IMB400 [18] visual sensor platform. This chapter is organized as follows. In Section 3.2 we investigate the dependency between the conditional entropy and the source rate. In Section 3.3 we use the conditional entropy to estimate source rates using two types of estimators: MAP and MMSE, and we show the performance of both estimators. In Section 3.4 we highlight the overhead of the feedback approach in a real visual sensor application, and we compare it

with the proposed estimation-based approach. In Section 3.5 we outline conclusions of this study.

3.2. Conditional Entropy-Source Rate Dependency

The conditional entropy $h_i = H(X_i \mid Y_i)$ evaluates the disparity between the bitplane being encoded (X_i) and the bitplane of the same significance in the key frame (Y_i). h_i provides the lower bound source rate for encoding X_i. In practice, the minimum source rate for encoding X_i that leads to error free decoding, denoted by r_i and referred to as the *ideal source rate*, will be greater than the h_i bound due to many reasons, such as the performance of the underlying channel code. The encoder can identify the value of r_i by following the feedback approach which is considered to be ideal in the sense that it will eventually lead to successful decoding for all encoded WZ frames while requiring the minimal number of parity bits. The estimation-based approach we are proposing aims to avoid feedback messages and incremental transmissions that could exhaust limited resources in visual sensor networks. At the same time, it aims to use source rates that are as close as possible to the ideal rates attained when feedback messages are exchanged. Our solution relies on estimating the ideal rate r_i given the value of the conditional entropy h_i. Efficient estimation is achievable if there is a high statistical dependency between the two variables (r_i and h_i). To examine the degree of this dependency, we conduct the following experiment:

Feedback scenario simulation experiment: We use our DVC codec described in Section 2.3 to encode a video sequence. For each pair of key and its following WZ frames, we find the conditional entropy for the i^{th} bitplanes $h_i = H(X_i \mid Y_i)$. Then starting from the lowest source rate we encode X_i and attempt to decode it at that rate. If the decoding fails, we move to the next higher source rate. We keep on increasing the source rate until X_i gets successfully decoded. At that point we record the source rate that led to successful decoding against the conditional entropy that we already have, i.e., we record the pair (h_i, r_i). Note that this simulates what happens in the feedback scenario. We repeat this experiment for all the bitplanes across all frames in the video sequence. When the encoder finishes encoding the whole video sequence, we will have an (h_i, r_i) pair related to each WZ frame.

Figure 3.1 shows the values of h_i and r_i related with the first 50 WZ frames of the Foreman video sequence (QCIF resolution). Both values are evaluated for i = 7 (the seventh bitplane). Note the resemblance in shape between the two curves. This suggests that the two values are correlated.

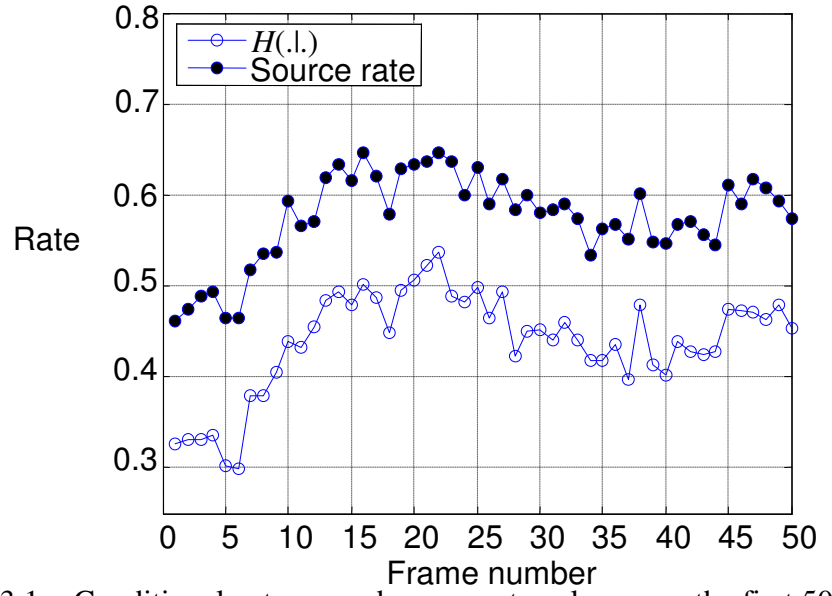


Figure 3.1: : Conditional entropy and source rate values over the first 50 WZ frames in the Foreman video sequence, seventh bit-plane.

To have a more concrete measure of statistical dependency, we treat h_i and r_i for the i^{th} bitplane as two random variables, and we compute the correlation coefficient [41]:

$$\rho_{h_i,r_i} = \frac{COV(h_i, r_i)}{\sigma_{h_i}\sigma_{r_i}} = \frac{E[h_i r_i] - E[h_i]E[r_i]}{\sigma_{h_i}\sigma_{r_i}} \qquad 3.1$$

Table 3.1 lists the correlation coefficient for different bitplanes in different video sequences (QCIF resolution Hall Monitor and Foreman). Note that the correlation coefficient is generally close to 1, which reflects the high correlation between the two values.

Bit-plane	Foreman	Hall Monitor
1	0.9871	0.9392
2	0.9948	0.9186
3	0.9885	0.8957
4	0.9635	0.8109
5	0.9118	0.925
6	0.9403	0.8284
7	0.97	0.952
8	0.9952	0.8246

Table 3.1: Correlation coefficients of different bit-planes in the QCIF resolution Foreman and Hall Monitor video sequences.

3.3. Source Rate Estimation and Performance

In this section we utilize the statistical dependency between the conditional entropy $h_i = H(X_i \mid Y_i)$ and the ideal source rate r_i and build estimators that map h_i to an estimate $\overline{r_i}$. We build two types of estimators:

(1) Maximum a Posteriori (MAP) estimator:

$$\overline{r_i} = F(h_i) = \arg\max_{r_i} p(r_i \mid h_i) \tag{3.2}$$

where $p(.|.)$ is the conditional probability mass function.

(2) Minimum Mean Squared Error (MMSE) estimator:

$$\overline{r_i} = F(h_i) = E[r_i \mid h_i] \tag{3.3}$$

Where $E[.|.]$ denotes the conditional expectation. Probability mass functions needed for both estimators are built offline (a training stage before the encoder starts encoding) from a large ensemble of (h_i, r_i) pairs generated by encoding a number of different video sequences. A source rate codebook is prepared and provided to the encoder. Each entry in the codebook has a conditional entropy value h_i and the corresponding estimated source

rate $\overline{r_i}$. The codebook has entries for all values of *i*. To evaluate the performance of both estimators, we use our encoder to encode different video sequences. For a pair of a key frame and its following WZ frame in a video sequence, the conditional entropy $h_i = H(X_i \mid Y_i)$ of the i^{th} bitplane is computed. The resulting h_i value is used to lookup the codebook. When a matching h_i value is found in the codebook, the corresponding $\overline{r_i}$ value is extracted and employed in encoding/decoding the bitplane. This is done for all values of *i*, i.e., for all the bitplanes that pertain to the WZ frame. When the bitplanes of this WZ frame are fully encoded, the number of bytes that resulted from compressing this frame is computed. On the other hand, when all the bitplanes that pertain to the WZ frame are fully decoded, they are merged into one frame, and the quality of that frame is evaluated using the Peak Signal to Noise Ratio (PSNR) measure. Finally, the total number of bytes that resulted from encoding all WZ frames in the video sequence as well as the average PSNR of those frames are computed and plotted.

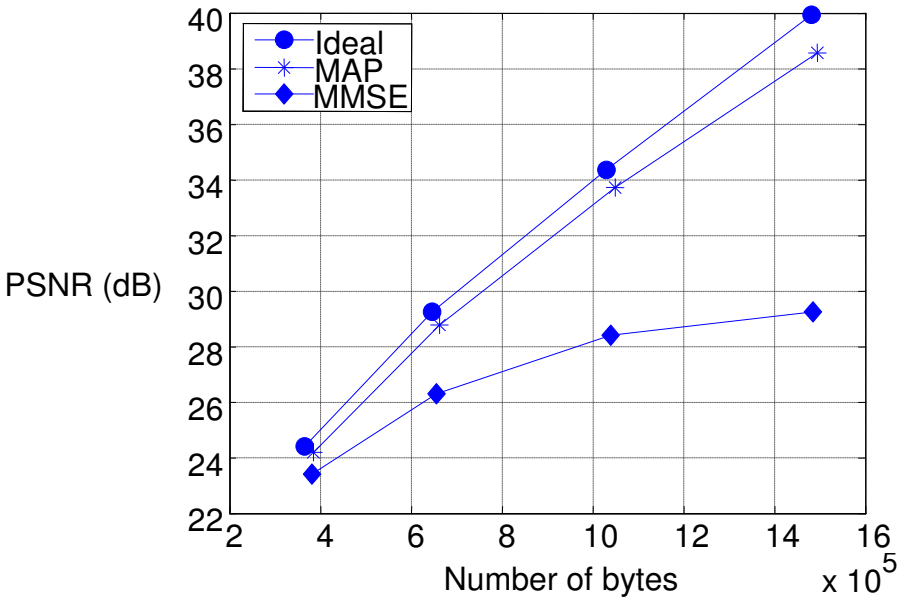


Figure 3.2: Performance using different methods to identify source rates - QCIF Forman video sequence.

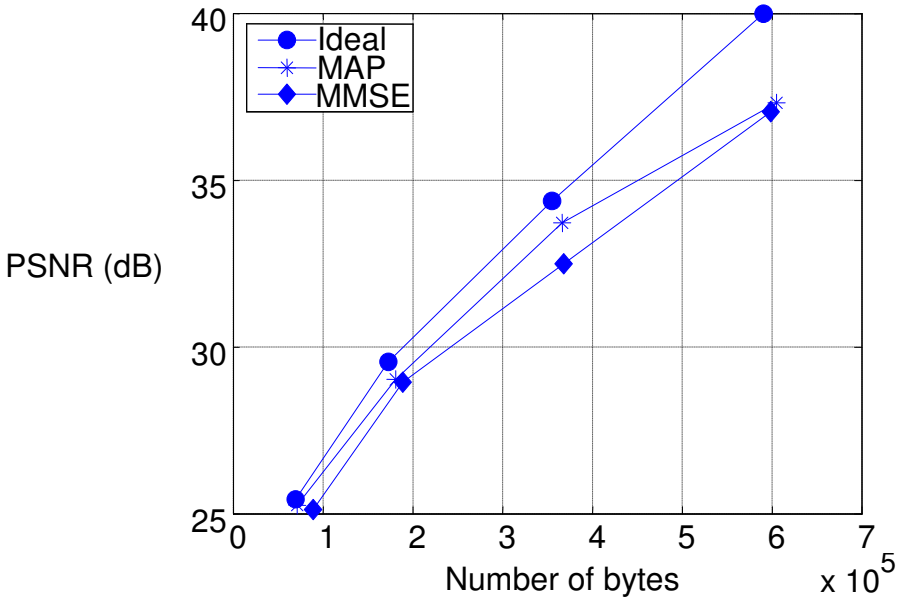


Figure 3.3: Performance using different methods to identify source rates - QCIF Salesman video sequence.

Figure 3.2 and Figure 3.3 plot the average PSNR vs. the total number of bytes for two different video sequences (QCIF resolution Foreman and Salesman). For each video

sequence we plot three curves; one for the feedback scenario where the minimal source rate r_i is reached in an incremental fashion, and the other two curves are related to the MAP and MMSE estimators. Each point in the figure corresponds to different quantization level, hence different number of bitplanes per frame. Note that for the Foreman sequence the MAP curve is close to the feedback scenario curve. For the Salesman sequence, both MAP and MMSE curves are close to the feedback scenario curve. This means that estimating source rates can achieve performance (in terms of compression and video quality) close to that of the feedback scenario. Further, we observe that, generally, for the two sequences, the MAP estimator outperforms its MMSE counterpart.

3.4. Overall Energy Consumption Performance

In this section we compare the performance (in terms of the overall energy consumption) of the feedback and estimation-based approaches when they are employed in a real visual sensor application. We implement and deploy the estimation-based DVC codec described in Section 2.3 on a real visual sensor platform that consists of the imote2 mote attached to the IMB400 multimedia board [18]. This platform is managed by the TinyOS operating system [28]. Further, we measured the energy consumption of radio, and the different computational modules in the imote2/IMB400 codec (DCT transform, parity-check matrix multiplication, etc.). In this section, we use our codec and the measured energy values (Section 2.4) to provide a comparison between the feedback and estimation-based approaches. We can group the differences between the two approaches into two categories:

1. Computation: Both approaches have exactly the same computational modules (DCT transform, parity-check matrix multiplication, etc.) with one exception; the estimation-based approach has one extra processing stage which is the conditional entropy computation. It is clear that the main part of this operation is computing the Hamming distance, which involves a set of XOR and addition operations. There are two features that make computing the Hamming distance on the imote2/IMB400 affordable: (1) XOR and addition operations are considered to be cheap in terms of energy consumption [33] compared to other types of operations (multiplication for instance). (2) Moreover, the cost of the Hamming distance computation is reduced by using the iwmmx [42] technology supported by imote2. This technology uses the SIMD (Single Instruction Multiple Data) processing paradigm that accelerates multimedia processing. Using SIMD, our codec performs 64 XOR operations in parallel. This shortens the execution time and hence energy consumption (energy is a function of execution time and power level). Those two features motivate using the conditional entropy as a basis to estimate source rates.

2. Transmission: There are three main differences between the two approaches in terms of transmission:

a) Ideal feedback approach results in more compression (Figure 3.2 and Figure 3.3) and hence fewer bits to transmit.

b) Ideal feedback approach suffers the cost of exchanging feedback messages.

c) In many occasions, the incremental transmission nature of the feedback approach results in transmitting more overhead data (packet headers) compared with the estimation-based approach. As the feedback approach transmits compressed data (parity

bits) in increments, this results in constraining the packet payload size to a certain value that is equal to the increment size. This may lead to transmitting compressed data of a certain frame through a large number of packets, and since each packet comprises a control header, this results in transmitting a large number of control data. On the other hand, the packet payload size in the estimation-based approach does not adhere to such constraint. Therefore, bigger payload sizes could be used, and hence compressed data could be communicated via smaller number of packets which means smaller number of control data. To further clarify this point, we use the following incremental transmission model: let the *increment size* Δ_i be the amount of compressed data (parity bits) belonging to the i^{th} bitplane that the encoder transmits in each transmission before waiting for a feedback. Consequently, in each transmission, the total amount of compressed data that the encoder transmits is $\Delta = \sum_{i=1}^{q} \Delta_i$, where q is the number of bitplanes that have not been completely communicated to the decoder yet. When the decoder acknowledges successful decoding for bitplane i, no more parity bits are needed (by the decoder) for that bitplane, hence the amount of transmitted data in the next transmission will decrease by Δ_i, and q will decrease by 1. Regarding feedback messages, the payload size is 1 byte. The i^{th} bit in this byte corresponds to the i^{th} bitplane, and the decoder sets this bit to one to acknowledge successful decoding for the corresponding bitplane, or to zero to request an increment of size Δ_i to be sent in the next transmission. Note that the maximum packet payload size in each transmission is Δ. In practice, control data overhead and feedback messages could severely degrade performance of the feedback approach. The following is an example that highlights this point.

Example: imote2 uses the CC2420 radio transceiver [21]. In TinyOS, each CC2420 packet has a header of 11 bytes. Moreover, our imote2/IMB400 codec employs an LDPCA code that supports 512 different rates to encode the 64×64 bitplanes. The minimum source rate is 1/512 and the maximum rate is 512/512 with an increment of 1/512. Note that the increment size $\Delta_i = (1/512) \times 4096 = 8$ bits (the same for all i), hence $\Delta = 8$ bytes. This means that, as long as the decoder has not acknowledged successful decoding for any bitplane, the encoder can send a packet with a maximum payload size of 8 bytes in each transmission.

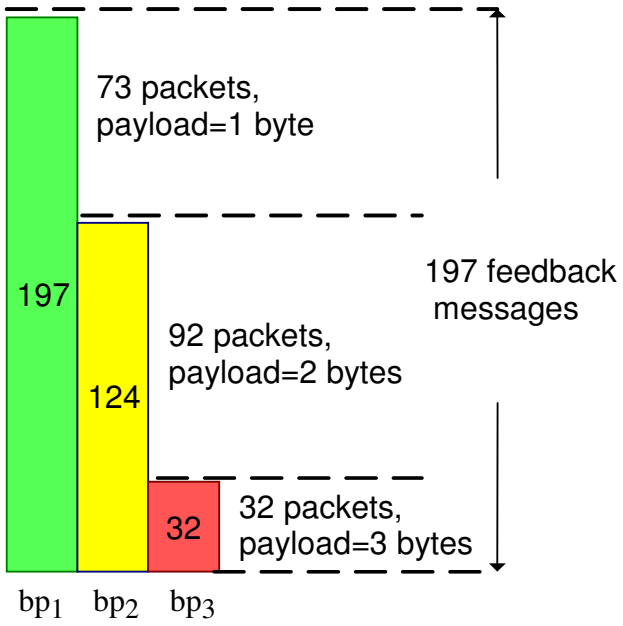


Figure 3.4: Incremental transmission example of three bitplanes.

Figure 3.4 shows a transmission status after bitplanes 4, 5, 6, 7, and 8 were acknowledged decoded. Currently we have $q = 3$ (3 bitplanes are not acknowledged yet). As the figure shows, at this point, bitplanes 1, 2, and 3 require 197, 124, and 32 more increments respectively to get decoded (the encoder does not know that, it keeps on sending increments until it gets back an acknowledgement). The payload in the next 32

packets will hold 3 bytes; each byte corresponds to one of the remaining bitplanes. After the encoder receives an acknowledgment that bitplane 3 is successfully decoded, bitplanes 1 and 2 will need to send 165 and 92 more increments respectively. Consequently, the next 92 packets will hold a payload of 2 bytes corresponding to bitplanes 1 and 2. After the encoder receives a feedback message indicating that bitplane 2 was successfully decoded, the next 73 packets will hold only one byte corresponding to bitplane 1. This procedure results in transmitting 197 packets with payload sizes that range from 1 to 3 bytes, in addition to 197 feedback messages. Note that, although the size of the compressed data in this example is 353 bytes, a total of 2167 control bytes (197 headers) were transmitted. This is an extremely large number compared with the compressed data size.

In the example above, if the source rates were known prior transmission, i.e., the encoder already knows that 197, 124, and 32 bytes would be sufficient to decode bitplanes 1, 2, and 3 respectively, then the 353 bytes of compressed data could be communicated to the decoder in a smaller number of packets as there will be no constraint on the payload size. For example, if we choose a payload size of 40 bytes, then we need only 9 packets to deliver the compressed data, and hence only 99 control bytes are transmitted. Since knowing the ideal rates prior transmission is not realistic, the proposed estimation-based approach provides the encoder with a mechanism to estimate those rates prior transmission, and hence this will eliminate the cost of overhead data. Despite this fact, the estimation-based approach suffers extra computation cost (conditional entropy computation), and it achieves less compression than the feedback approach (larger volume of compressed data). Therefore, we need to compare between

the two approaches in terms of the *overall* energy consumption (computation and transmission). To do that, we use our estimation-based imote2/IMB400 codec to encode the Foreman and the Salesman video sequences. We choose the packet payload size to be 40 bytes (remember that the header size in TinyOS is 11 bytes), and a MAP estimator to estimate rates. Further, we simulate the feedback approach to compress the same video sequences in the same manner described in Section 3.3 (but now the video sequences are down sampled to a 64×64 resolution to be consistent with the imote2/IMB400 decoder). The LDPCA code used here is the same as in the previous example (increments of 1/512). For energy measurement, we use the following values [16]: (1) cost of transmitting/receiving 1 byte is $66.67\,\mu J$, and (2) cost of conditional entropy computation is $2.7681\,mJ$ per frame (8 bitplanes).

Figure 3.5 and Figure 3.6 demonstrate the performance of each approach in terms of energy consumption and video quality at different quantization levels. In this figure, energy consumption of the computation modules that are common between the two scenarios (such as DCT, parity-check matrix multiplication, etc.) is not considered. Note that there are two curves related to the feedback approach; one curve (feedback1 in the figure) corresponds to a scenario where a feedback message is sent back to the encoder after each transmission. The other curve (feedback2 in the figure) assumes that the decoder sends a feedback only to acknowledge successful decoding of any bitplane. Hence, after each transmission, the encoder waits for a period of time for an acknowledgement. If no acknowledgement is received, then the encoder assumes that the decoder failed to decode all bitplanes, and it transmits a new increment. Note that although this feedback scheme results in less frequent feedbacks, the estimation-based

approach demonstrates lower energy consumption for a wide range of video quality levels.

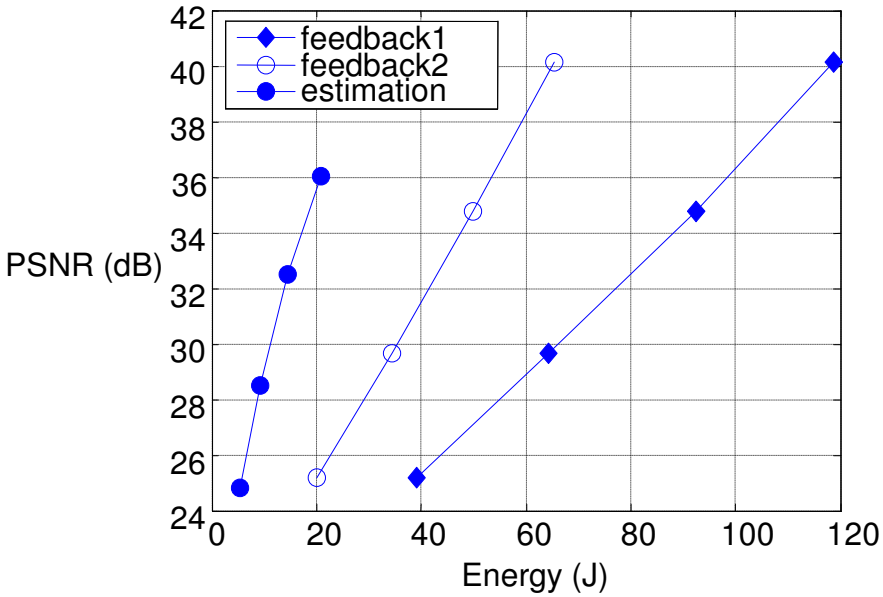


Figure 3.5: Performance over a real visual platform - Forman video sequence.

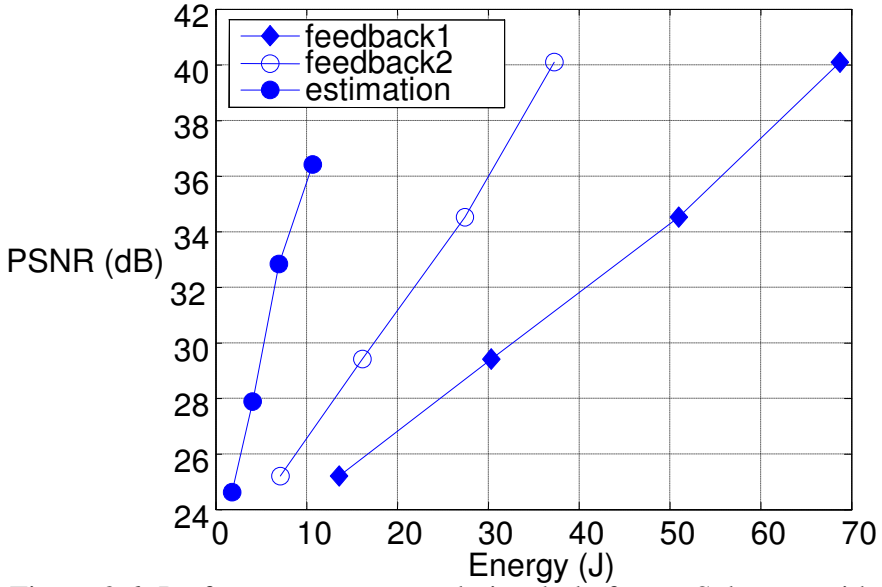


Figure 3.6: Performance over a real visual platform - Salesman video sequence.

3.5. Discussion

In this chapter we showed that the statistical dependency between the conditional entropy (entropy of a WZ source conditioned on a side information source) and the ideal source rate of encoding a WZ frame (the minimal rate that leads to successful decoding) can be utilized to estimate source rates, and hence avoid exchanging feedback messages with the decoder. We showed that the estimation-based approach results in a performance that is close to that of the feedback approach in terms of compression and video quality. Further, we showed that the estimation based approach can demonstrate lower energy consumption for a range of video quality levels compared with the feedback approach when they are deployed over a real visual sensor platform (imote2/IMB400).

Chapter 4

Polar Codes for Low Resolution Distributed Video Coding

Polar codes are the first codes proven to achieve capacity while having low encoding and decoding complexity. As the development of polar codes is relatively recent, there has been little work to assess the potential of polar codes for practical applications. In this chapter, we employ and evaluate the performance of polar codes in Distributed Video Coding (DVC). Further, we compare the performance of polar codes with the more established and more investigated Low Density Parity Check Accumulate (LDPCA) codes in the context of DVC. Our study shows: (a) Polar and LDPCA codes exhibit almost similar performance for DVC coding of large image blocks. (b) Polar codes offer a clear advantage when coding smaller size image blocks. Consequently, polar codes could represent a viable solution for distributed sensor networks that capture a low-resolution video signal at each visual sensor node.

4.1. Introduction

Similar to previous chapters, the DVC that we investigate in this chapter is based on *asymmetric* Distributed Source Coding (DSC), which is a special case of the broader DSC problem [1]. Let X and Y be two sources that are statistically dependent. Y, referred to as a *side information* (or key) source, is intra-coded and communicated to the decoder at rate $R_Y \geq H(Y)$. By utilizing the statistical dependency between the two sources at the decoder side, it is shown in [1] that X, referred to as a *Wyner-Ziv* (WZ) frame, can be compressed with no loss, and with having no access to Y at rate $R_X \geq H(X \mid Y)$. Interestingly, the aggregate rate $R_X + R_Y$ can achieve the joint entropy $H(X,Y)$. Hence, DVC can achieve the same rate as the computationally demanding joint encoding (requires access to both X and Y) by performing low complexity independent encoding (no access to Y). Due to the low complexity encoding, DVC is considered to be suitable for visual sensor networking applications [43] where visual sensors are limited in their resources (computational power, storage, etc.).

Channel codes are heavily employed in DVC. The reason is that the disparity between the two sources, X and Y, can be viewed as noise. In other words, X can be viewed as a noisy version of Y. Consequently, if Y is already communicated to the decoder, then the encoder can communicate X by transmitting a number of parity bits sufficient to correct the noise in Y. Compression is achieved by transmitting the parity bits. The level of compression of DVC depends primarily on the underlying channel code. Therefore, it is common to employ *capacity-approaching* codes. Examples of DVC schemes that employ such codes can be found in [44] where Turbo codes are used, and in [10] where LDPC

codes are used. Although these codes and others are known to be capacity-approaching, there are no formal proofs on their optimality. Polar codes, recently introduced by Arikan [7], are the first channel codes proven to achieve capacity for many types of channels. Meanwhile, they are described to be practical in terms of their encoding/decoding complexity. Due to their interesting features, polar codes can be utilized in achieving optimal performance in many problems in the field of information theory. For example, in [45], polar codes were constructed to achieve optimal performance in lossy and lossless source compression, in addition to the Wyner-Ziv, and Gelfand-Pinsker coding problems.

Both polar and LDPC codes have some attributes that qualify them to be employed in DVC over resource-limited visual sensor networks (which is one of the main applications of DVC). For example, (1) the parity check matrix of LDPC codes and the generator matrix of polar codes are sparse, which allows for compact storage. (2) Both codes have low encoding complexity ($O(N)$ for LDPC codes and $O(N \log N)$ for polar codes where N is the code block length). (3) Both codes can operate in a rate-adaptive mode where the encoder transmits compressed data (parity bits) in increments until the decoder signals successful decoding. This is important in situations where the encoder lacks knowledge about the disparity (correlation channel) between the WZ and the key sources.

It is well known that polar and LDPC codes are powerful under asymptotic conditions, i.e., as N (code block length) goes to infinity. However, in real visual sensor applications, codes are usually applied on short blocks since many visual sensors are endowed with low resolution imagers. For example, Cyclops [17] captures 128×128 pixel images and both WiSN [46] and MeshEye [47] support capturing of images with 30×30

pixel resolution. In this chapter we demonstrate the practical utility of polar codes in the context of low resolution (and hence short block length) rate-adaptive DVC. More specifically, we demonstrate the performance advantages of polar codes over the more investigated LDPCA codes [36] (rate adaptive LDPC codes that are designed for DSC) in the context of low resolution DVC. To that end, we implement a DVC codec that employs both codes. We apply the codec on a set of video sequences with different (low) resolutions. To our knowledge, this is the first work that incorporates polar codes in a real DVC application (as opposed to LDPCA codes). This chapter is organized as follows. In Section 4.2, we provide a brief description of polar codes and the DSC scheme that we employ in our DVC codec. In Section 4.3, we describe our rate-adaptive DVC codec then we present our experimental results. In Section 4.4, we outline main conclusions.

4.2. Rate-adaptive Polar Codes for DSC

In this section, we describe in detail a rate-adaptive DSC scheme that incorporates polar codes. In the next section we will use this scheme to implement DVC over low resolution images. For a detailed description of LDPCA codes refer to APPENDIX A, and for a detailed description of our LDPCA DSC scheme refer to Chapter 3.

4.2.1. Polar Codes

Let X_0^{N-1} denote the random vector $[X_0,...X_{N-1}]$. If F is a subset of $\{0,...,N-1\}$, then F^c is the complement of F. $(X_0^{N-1})_F$ denotes elements of X_0^{N-1} indexed by

members of F. Ber(p) denotes a Bernoulli random variable with parameter p, and $h_2(p)$ is the corresponding binary entropy. Let W be a Binary-input Discrete Memoryless Chanel (B-DMC) with transition probability $W(y|u)$. In this chapter, we are interested in the case where $\{0,1\}$ is the alphabet for both the input and the output of W. Let W^N be a compound channel that consists of $N = 2^n$ independent copies of W. The transition probability of W^N is:

$$W^N(y_0^{N-1}|u_0^{N-1}) = \prod_{i=0}^{N-1} W(y_i|u_i) \tag{4.1}$$

Polar coding places a pre-processing stage before transmitting u_0^{N-1} over W^N:

$$x_0^{N-1} = u_0^{N-1} G_2^{\otimes n} \tag{4.2}$$

where G_2 is the polarization matrix defined as:

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$\otimes$ is the n^{th} Kronecker product operator . This pre-processing operation results into a new compound channel W_N that consists of N *dependent* channels. The transition probability of the i^{th} channel is:

$$W_N^{(i)}(y_0^{N-1}, u_0^{i-1}|u_i) = \frac{1}{2^{N-1}} \sum_{u_{i+1}^{N-1}} W_N(y_0^{N-1}|u_0^{N-1}) \tag{4.3}$$

where

$$W_N(y_0^{N-1}|u_0^{N-1}) = W^N(y_0^{N-1}|u_0^{N-1}G_2^{\otimes n}) \tag{4.4}$$

The Successive Cancelation (SC) algorithm is used for decoding polar codes where Equation 4.3 is applied iteratively. At the i^{th} iteration, u_0^{i-1} that was estimated over the

earlier *i*-1 iterations is used to estimate u_i. As shown in [7], asymptotically (as *N* goes to infinity), a group of the dependent channels, $W_N^{(i)}$, become completely reliable, i.e., the input of the channel can be estimated with small probability of error. The fraction of reliable channels converges to the capacity of *W*. On the other hand, the remaining channels become completely unreliable. This phenomenon is known as *channel polarization*.

Polar coding: Let A and A^c be the set of indices that correspond to the reliable and unreliable channels respectively. We will refer to A as the *information set*, and A^c as the *frozen set*. To map a data vector d into a codeword x_0^{N-1}, we set u_0^{N-1} as follows: $(u_0^{N-1})_A = d$, and $(u_0^{N-1})_{A^c} = f$, where f is a vector of bits that is fixed across all the codewords. Next, u_0^{N-1} is multiplied by $G_2^{\otimes n}$ which results in the codeword x_0^{N-1}. f and A^c are considered to be parameters of the code that are known to both the encoder and decoder.

4.2.2. DSC with Polar Codes

The basic idea of employing polar codes in DVC is described in [48]. It is based on the fact that X (WZ frame) can be regarded as a noisy version of Y (key frame). i.e., $X = Y + Z$, where Z is a Ber(*p*) noise sequence (we consider the correlation channel as a Binary Symmetric Channel with parameter *p*; BSC(*p*)). Y is assumed to be available at

the decoder side. Our goal is to communicate X at a rate close to the theoretical bound. In other words, we aim to encode Z. At the encoder side, X is multiplied by $G_2^{\otimes n}$:

$$X_0^{N-1}G_2^{\otimes n} = (Y_0^{N-1} + Z_0^{N-1})G_2^{\otimes n}$$

$$= Y_0^{N-1}G_2^{\otimes n} + Z_0^{N-1}G_2^{\otimes n} \quad 4.5$$

Note that $Y_0^{N-1}G_2^{\otimes n}$ in Equation 4.5 is already available at the decoder. Therefore, the encoder needs to communicate $Z_0^{N-1}G_2^{\otimes n}$ to enable the decoder reconstructing $X_0^{N-1}G_2^{\otimes n}$ according to (4.5). Let $U_0^{N-1} = Z_0^{N-1}G_2^{\otimes n}$. Note that U_0^{N-1} is the polarized noise sequence. The Entropy rate of U_0^{N-1} is:

$$H(U_0^{N-1}) = nh_2(p) = \sum_{i=0}^{N-1} H(U_i \mid U_0^{i-1}) \quad 4.6$$

In the context of source coding, polarizing a bit sequence results into two groups of bits: (1) a group of reliable bits indexed by the information set where the conditional entropy in (6) is close to 0. This means that given U_0^{i-1}, U_i can be estimated reliably. (2) A group of unreliable bits that are indexed by the frozen set where the conditional entropy is close to 1. Let A and A^c denote the information and frozen sets respectively. It is clear from (6) that the fraction of frozen bits, $(U_0^{N-1})_{A^c}$, is $h_2(p)$. Therefore, if the encoder transmits the frozen bits $(U_0^{N-1})_{A^c}$, the decoder will be able to estimate $(U_0^{N-1})_A$, and the source rate would be close to the theoretical bound $h_2(p)$.

Transmitting $(Z_0^{N-1}G_2^{\otimes n})_{A^c}$ to the decoder requires separating between Z and Y at the encoder side, which, in practice, is not possible. Therefore, the encoder transmits

$(X_0^{N-1}G_2^{\otimes n})_{A^c} = (Y_0^{N-1}G_2^{\otimes n} + Z_0^{N-1}G_2^{\otimes n})_{A^c}$. As the decoder knows A^c and already has $Y_0^{N-1}G_2^{\otimes n}$, it simply extracts $(Z_0^{N-1}G_2^{\otimes n})_{A^c}$ as follows:

$$(Y_0^{N-1}G_2^{\otimes n} + Z_0^{N-1}G_2^{\otimes n})_{A^c} + (Y_0^{N-1}G_2^{\otimes n})_{A^c} = (Z_0^{N-1}G_2^{\otimes n})_{A^c} \quad 4.7$$

Once the decoder obtains $(Z_0^{N-1}G_2^{\otimes n})_{A^c}$, it uses it to estimate $(Z_0^{N-1}G_2^{\otimes n})_A$ using Equation 4.3 after setting the output of the channel to a vector of zeros:

$$W_N^{(i)}(u_0^{i-1} \mid u_i) = W_N^{(i)}(y_0^{N-1} = \bar{0}, u_0^{i-1} \mid u_i) \quad 4.8$$

Next, the decoder combines the information and frozen bits in one vector $Z_0^{N-1}G_2^{\otimes n}$. Finally, polarization is inversed:

$$Z_0^{N-1} = (Z_0^{N-1}G_2^{\otimes n})G_2^{\otimes n} \quad 4.9$$

And this sequence is used to get X back:

$$X_0^{N-1} = Y_0^{N-1} + Z_0^{N-1} \quad 4.10$$

Frozen and information set construction: as mentioned earlier, we consider the correlation channel as being a BSC(p). For a certain value of p, A and A^c are constructed in an offline training stage as follows: a sequence Z of length N is generated according to a Ber(p) random variable. Z is then polarized; $U_0^{N-1} = Z_0^{N-1}G_2^{\otimes n}$. The SC algorithm uses (8) to generate an estimate $\bar{U}_i$ of the polarized bits for $i = 0...N-1$. This is repeated for a large number of times in order to construct a joint probability density function (pdf) of U_i and $\bar{U}_i$. This pdf is used to

compute the mutual information $I(U_i;\bar{U}_i)$. Values of i where the mutual information is close to 1 is assigned to A, whereas values of i with mutual information close to 0 is assigned to A^c.

Rate adaptation: In some situations where the encoder is unaware of the correlation channel parameter *p*, it is useful to transmit frozen bits in increments, starting from a small number of bits until the decoder signals successful decoding. The worst case would be transmitting the whole polarized WZ bit sequence; $X_0^{N-1}G_2^{\otimes n}$. In this case, all bits would be treated as frozen bits. Therefore, to achieve rate adaptation, we treat the *N* bits of $X_0^{N-1}G_2^{\otimes n}$ as frozen bits. Meanwhile, we rank those frozen bits to indicate which ones would be transmitted first. To achieve that, we conduct the training process described earlier a certain number of times starting with a small value of *p*, and we gradually increase *p* until it reaches its maximum value (0.5). Note that each training run results in different A and A^c. As p gets larger, some indices that were in A in the previous training run will move to A^c in the current training run (as the capacity of the channel decreases). Indices that move in earlier training runs (runs with smaller *p*) are considered less reliable, and they are transmitted in earlier increments. This way we are able to rank the *N* bits in terms of their transmission precedence.

4.3. Experimental Results

4.3.1. DVC Codec

The following is a brief description of the DVC codec that we use in our experiments. Frames are processed in pairs. In each pair, the first frame is a Key frame, and the second frame is a WZ frame. A key frame Y is intra-coded and communicated to the decoder. A WZ frame X is transformed using a 4×4 Discrete Cosine Transform (DCT). A uniform quantizer of $M = 2^b$ intervals is used to quantize the DCT coefficients. Quantized coefficients are then represented via Gray codes (as opposed to regular binary representation). Coefficients of the same frequency are grouped together to form *frequency bands*. Each band is processed separately as follows: First, it is decomposed into b bitplanes. Next, Polar codes are used to encode each bitplane as described in Section 4.2. The encoder transmits bits incrementally starting with bits that are ranked as least reliable. The corresponding bitplane in Y is used to extract and decode the polarized noise sequence. We log the minimum source rate that results in error free decoding. To compare with LDPCA, we repeat the same experiments using our DVC codec modified to use LDPCA codes.

4.3.2. Results

Figure 4.1 demonstrates the performance of the polar and LDPCA codes when applied to the Akiyo video sequence with different resolutions. Each point in the figure corresponds to a compression ratio and the corresponding Peak Signal to Noise Ratio (PSNR). Different points correspond to different DCT quantization levels. The compression ratio is defined as the total number of bytes after compression divided by the total number of bytes before compression. We use two different LDPCA codes whose performances were reported in [36]. The first one is regular with degree distribution

given by λ_3=1, where λ_r is the proportion of variable nodes with degree *r*.The second is irregular with degree distribution (λ_2=0.316, λ_3=0.415, λ_7=0.128 λ_8=0.069, λ_{19}=0.02, λ_{21}=0.052). Figure 4.1 (a) shows the performance for the 128×128 resolution case (block length equals 1024 as each of the 16 DCT coefficient bands is encoded separately). The figure shows that generally the three codes are achieving almost the same compression performance. Irregular LDPCA slightly outperforms the regular LDPCA and polar codes. Figure 1 (b) shows the performance for the 64×64 case (block length is 256). Interestingly, at this block length, polar code achieves more compression compared to LDPCA codes. As the resolution goes down to 32×32 (Figure 1 (c) where block length=64), not only polar codes outperform both LDPCA codes, but also the distance between the performance curves become longer. This suggests that as the block length goes down, polar codes demonstrate more compression compared to LDPCA codes. To support this claim, we repeat our experiment using more video sequences. We use the Hall-monitor, Salesman, Foreman, and Carphone video sequences. The results for the aforementioned video sequences are shown in are shown in Figure 4.2, Figure 4.3, Figure 4.4, and Figure 4.5. By looking at the figures we can notice a trend; for the 128×128 case, polar and LDPCA codes behave closely. As the resolution goes down, the performance gap becomes larger where polar codes achieve more compression. This trend can be seen clearly in Figure 4.6 that averages the results over the different video sequences.

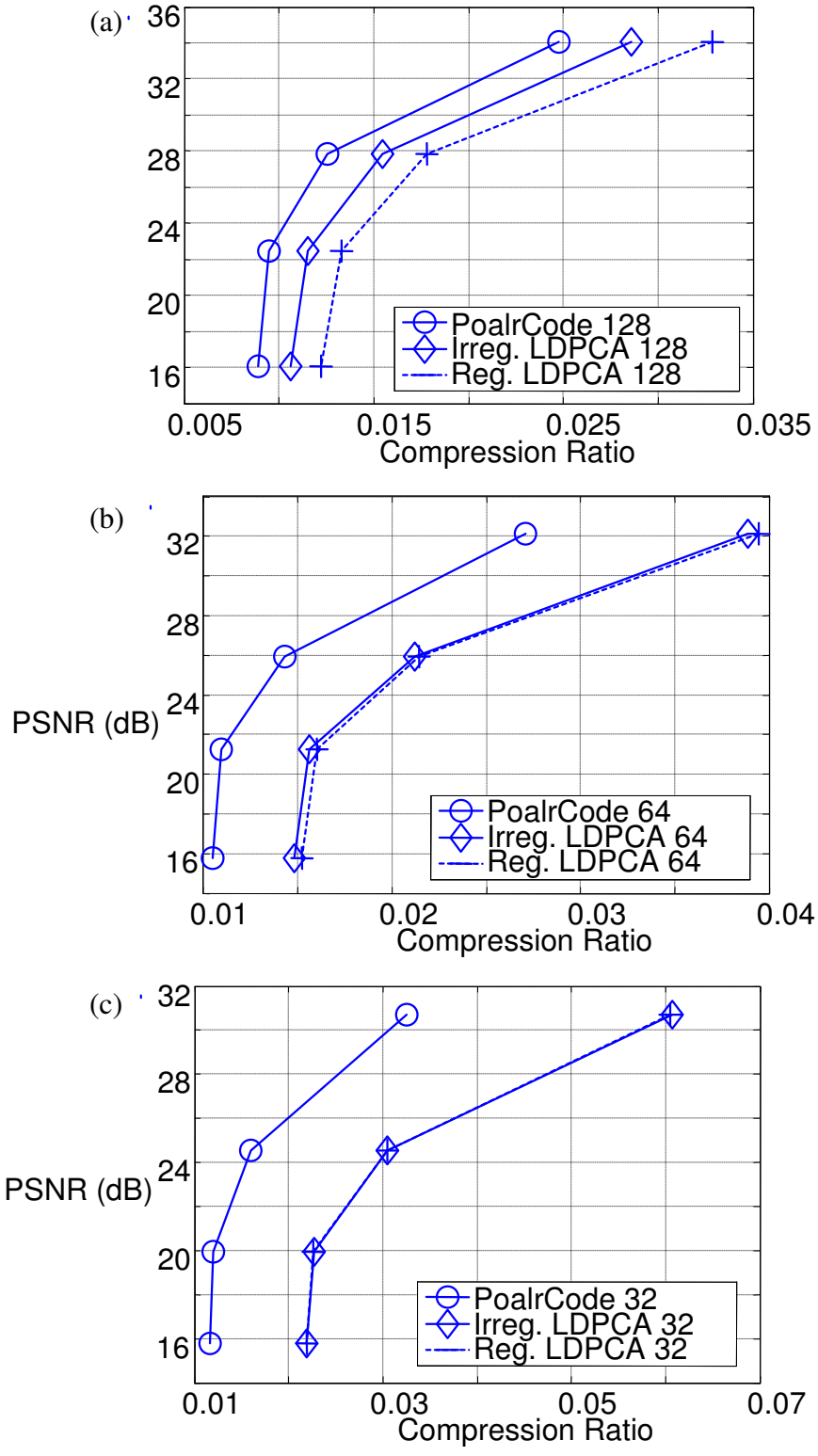


Figure 4.1: Performance of DVC using polar, regular and irregular LDPCA codes applied on (a) 128×128 Akiyo, (b) 64×64 Akiyo, and (c) 32×32 Akyio.

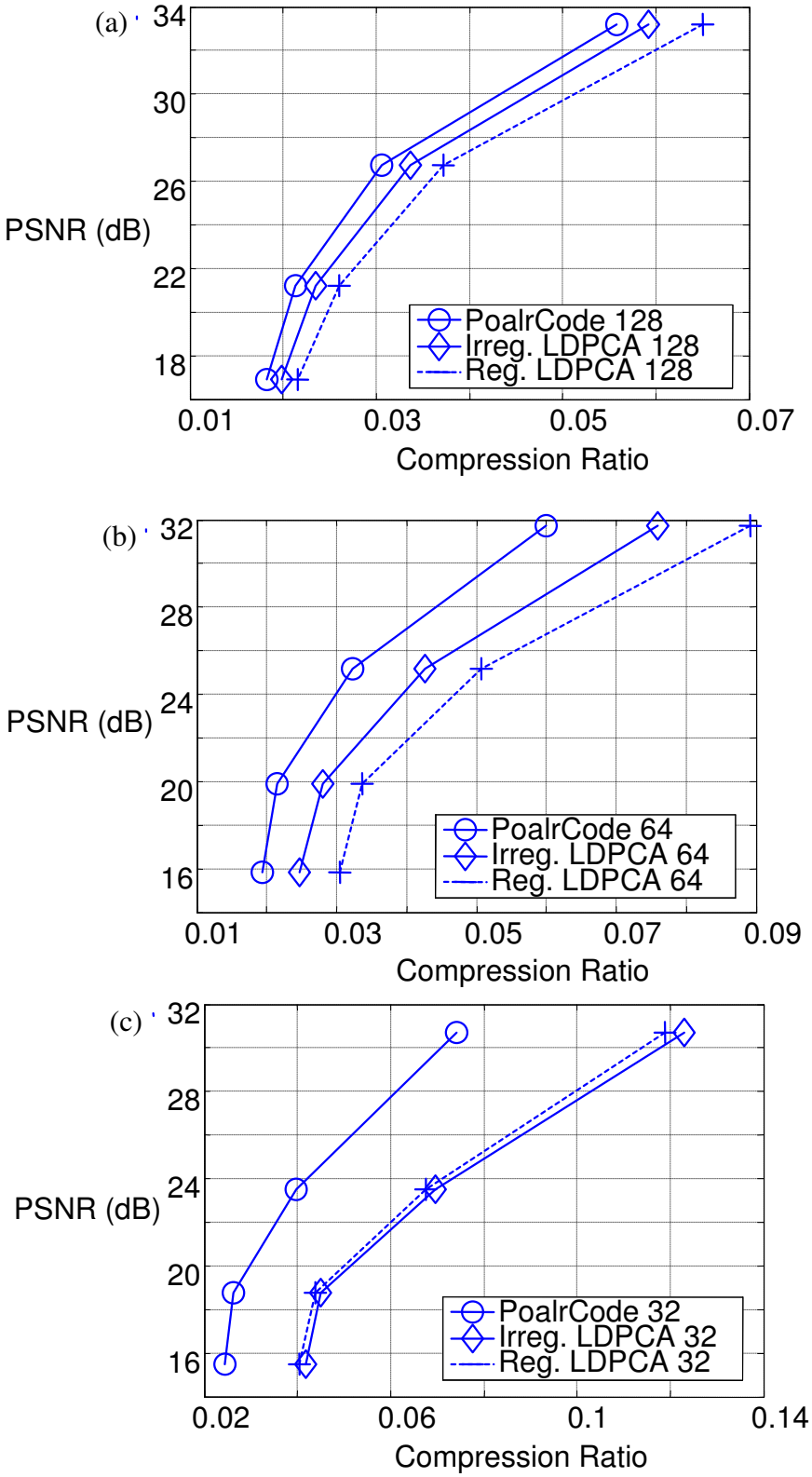


Figure 4.2: Performance of DVC using polar, regular and irregular LDPCA codes applied on (a) 128×128 Hall-monitor, (b) 64×64 Hall-monitor, and (c) 32×32 Hall-monitor.

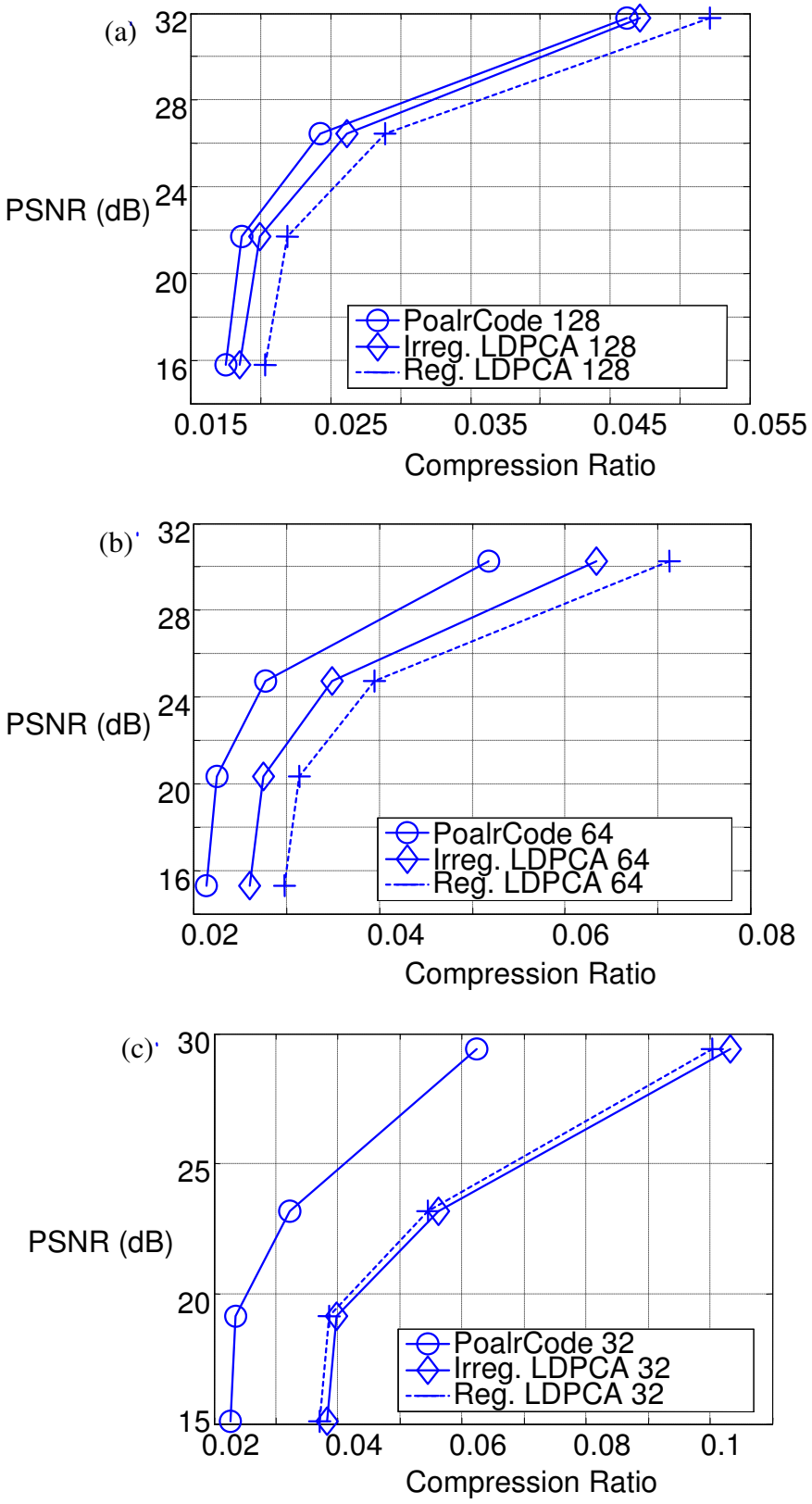


Figure 4.3: Performance of DVC using polar, regular and irregular LDPCA codes applied on (a) 128×128 Salesman, (b) 64×64 Salesman, and (c) 32×32 Salesman.

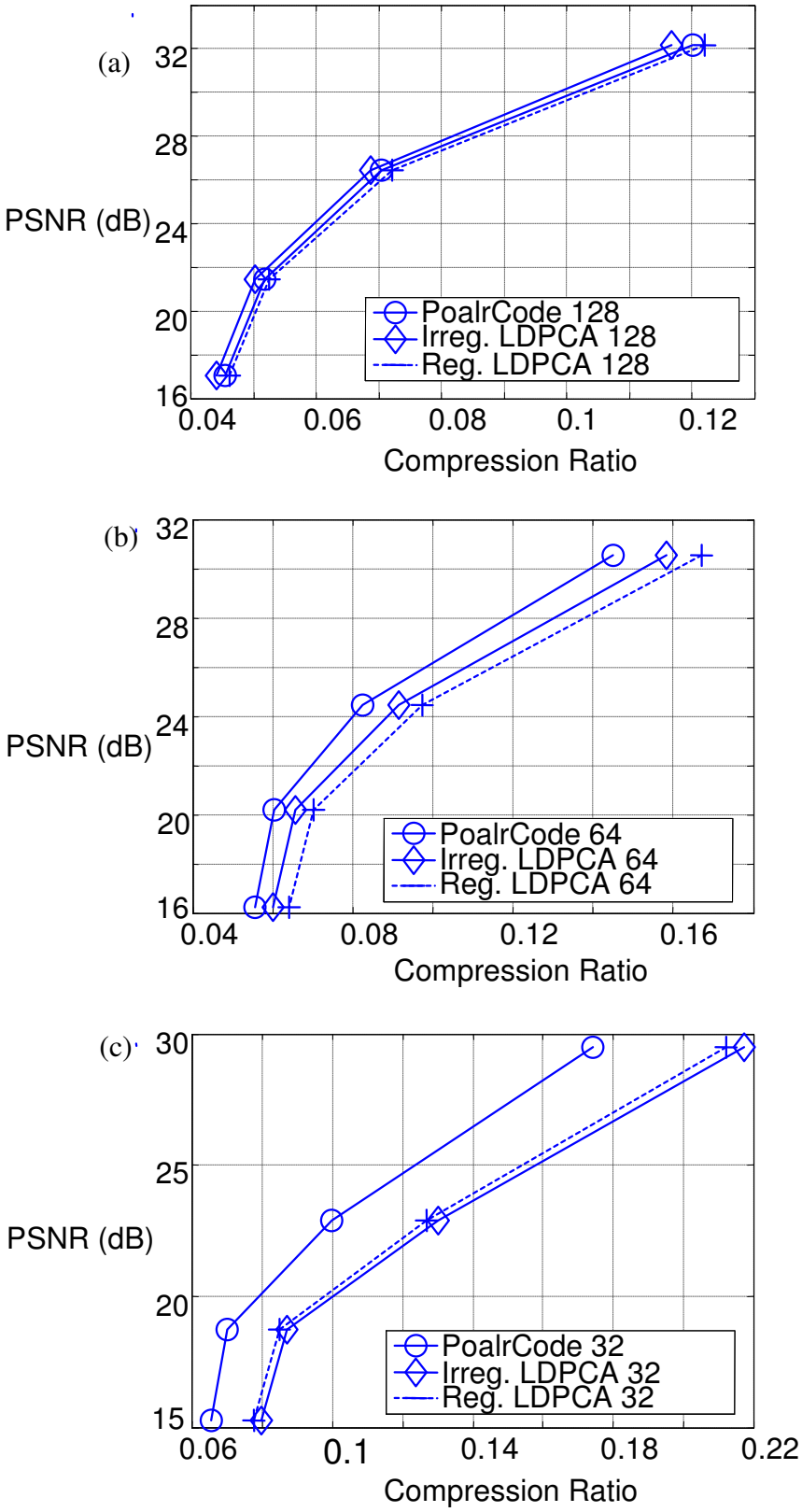


Figure 4.4: Performance of DVC using polar, regular and irregular LDPCA codes applied on (a) 128×128 Foreman, (b) 64×64 Foreman, and (c) 32×32 Foreman.

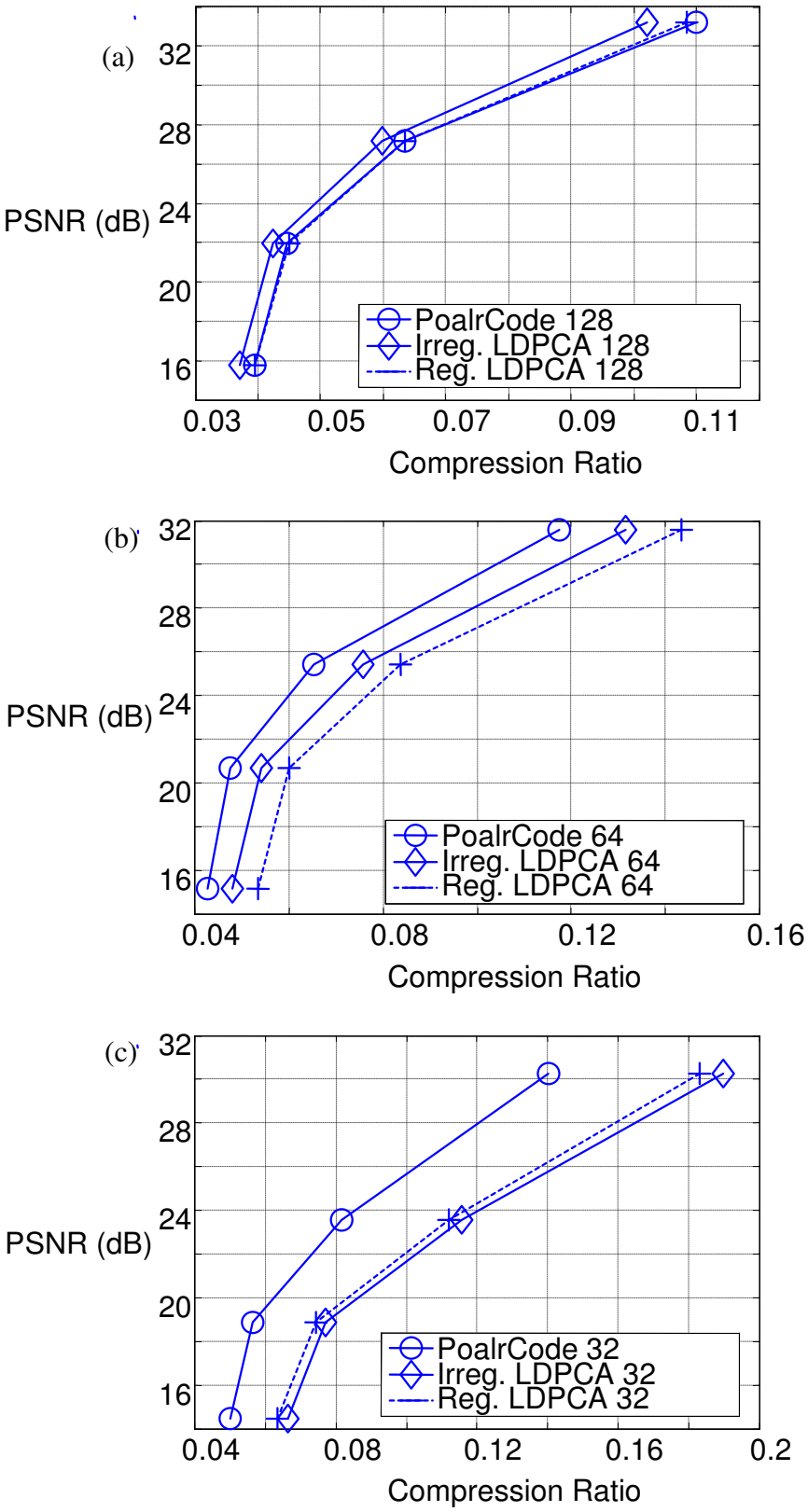


Figure 4.5: Performance of DVC using polar, regular and irregular LDPCA codes applied on (a) 128×128 Carphone, (b) 64×64 Carphone, and (c) 32×32 Carphone.

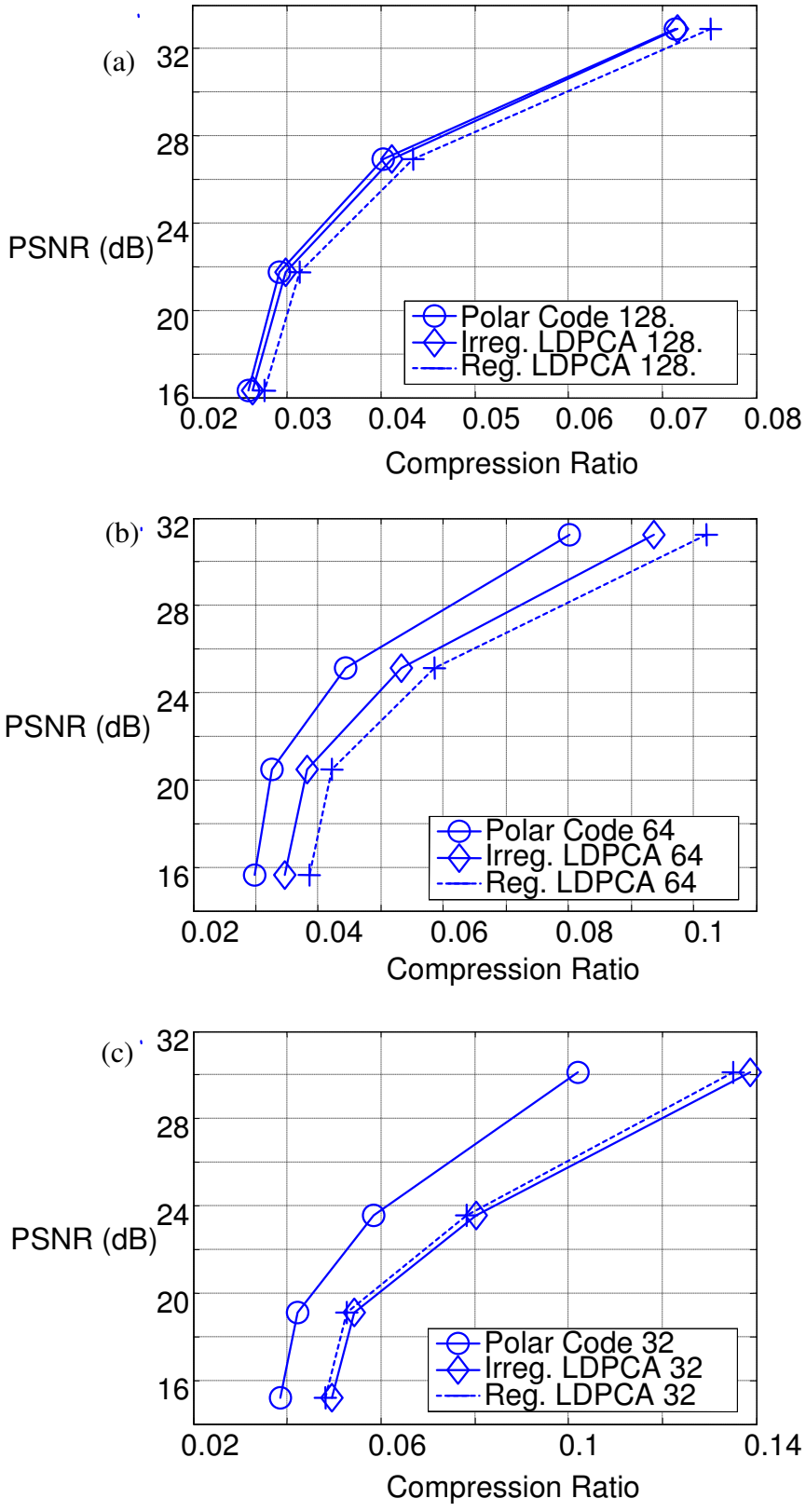


Figure 4.6: Performance averaged over the five video sequences with (a) 128×128, (b) 64×64 and (c) 32×32 resolutions.

4.4. Discussion

In this chapter we build a rate-adaptive DVC codec that incorporates polar and LDPCA codes. We apply our codec on a set of video sequences with different (low) resolutions. We show that Polar and LDPCA codes exhibit almost similar compression performance for DVC coding with large blocks. We also show that polar codes offer a clear advantage when coding smaller size image blocks.

Chapter 5

HopCaster: A Network Coding-Based Hop-by-Hop Reliable Multicast Protocol

Intra-flow Network Coding (NC) is an emerging technique that has potential in significantly improving the performance of multicasting over Wireless Mesh Networks (WMNs). It achieves that by allowing intermediate forwarding nodes (FNs) to utilize overhearing and coding to reduce the total number of transmissions required for end-to-end data delivery. The existing intra-flow NC-based multicast protocols are all based on the conventional end-to-end transport principle. A major downside for such principal is that intermediate FNs are not able to accurately determine the minimum number of coded packets they should transmit in order to ensure successful data delivery to the destinations, and hence redundant packets can be injected into the network, leading to significant performance degradation. Furthermore, existing protocols do not accomplish fairness between multicast receivers with good network connectivity and those with bad connectivity (heterogeneous receivers).

We argue that a receiver-driven hop-by-hop transport approach demonstrates a higher performance compared to the end-to-end scheme when incorporated in intra-flow NC-based multicasting. The hop-by-hop transport approach allows exploiting not only computation power (in performing network coding) but also in-network storage (in doing hop-by-hop caching) at the intermediate FNs to achieve performance gain. In this chapter we propose HopCaster, a novel protocol that incorporates intra-flow NC with hop-by-hop transport to achieve high-throughput reliable multicasting and to solve receiver heterogeneity issues. It completely eliminates the need for estimating the number of coded packets to be transmitted by a FN. It also simplifies multicast management and congestion control. Moreover, HopCaster employs a cross-layer rate adaptation mechanism that optimizes the multicast data rate in hop-by-hop transport by exploiting the receiver population changes over time. We compare the hop-by-hop and end-to-end transport approaches. Our evaluations show that, compared to Pacifier, a state-of-the-art end-to-end intra-flow NC-based multicasting protocol, HopCaster achieves up to 29% throughput improvement.

5.1. Introduction

Reliable multicast applications such as, file sharing, data casting, software upgrade, etc. are becoming increasingly common. These applications not only have a strict reliability requirement of 100% packet delivery ratio (since any loss causes quality degradation) but also require high throughput (as fast download is always desirable from a user standpoint). Achieving (1) reliability and (2) high-throughput in multicasting applications over multi-hop wireless mesh networks (WMNs) faces many challenges.

One is imposed by the lossy and time-varying nature of wireless transmission environments, which necessitates employing adaptive mechanisms to ensure reliability and high throughput. Another main challenge, which is unique to multicasting, is the bandwidth heterogeneity amongst multicast receivers. The receivers with poor connectivity and low-throughput paths to the source may greatly degrade the performance of receivers with good connectivity conditions.

Traditional multicasting protocols that were devised to achieve reliability and high-throughput are client-server based where intermediate routers or forwarding nodes (FNs) simply duplicate and forward packets. Many of these protocols employ end-to-end forward error correction (FEC), automatic repeat request (ARQ) or hybrid FEC-ARQ techniques to achieve reliability in wireless networks [49] [50] [51] [52] [53]. In many cases the performance is limited by the multicast destination receivers with the worst path to the source.

Network coding (NC) is emerging as an innovative technique to improve reliability and throughput in WMNs. NC takes advantage of the broadcast nature of the wireless medium by allowing intermediate FNs to encode received and overheard data packets (instead of simply duplicate and forward packets) [4] [6] [54] [55] [56] [57] [58] [59]. NC can be classified as (1) inter-flow NC and (2) intra-flow NC. With inter-flow NC, an intermediate FN can encode/forward data packets belonging to different flows. A receiver decodes the packets to obtain the flow of data targeted to it using its knowledge of another flow. However in order to obtain inter-flow coding opportunities, the encoded flows need to pass through a common FN with certain network topologies, for example, “Alice-Bob structure” or X-structure” [55] [56] [59]. Furthermore, inter-flow NC requires

that the sending FN knows what data packets have been buffered or overheard by each of the intended receivers in order to determine how to encode the packets across the flows. This thereby leads to increased control overhead.

More recently, intra-flow NC based on random linear block codes [6] [54] has attracted research attention due to its simplicity and efficiency. In intra-flow NC, a FN generates and forwards random linear combinations of received (and overheard) packets belonging to a flow over a certain Galois field. The random mixing at each FN ensures that if a group of FNs heard the same packet transmission, the coded packets that they generate and forward are, with high probability, linearly independent. This reduces duplicate packet transmissions over the shared wireless medium making opportunistic overhearing and forwarding more effective in WMNs [57]. A node can receive, in addition to coded packets transmitted by its direct parent node, coded packets of the same flow transmitted by other neighbors, achieving significant performance gains. Compared to inter-flow NC, intra-flow NC has the advantages of low signaling overhead and ease of implementation with no topological requirements..

The use of intra-flow NC introduces new challenges in designing a practical multicasting protocol. (1) First, when a FN receives a coded packet, it has to generate (and then forward) a new coded packet by performing random mixing on the previously received and cached packets and the most recently received one. To combat the lossy nature of wireless channels, for one received packet, the coding and forwarding has to be repeated multiple time (each time with different random mixing coefficients) in order to ensure that at least one of the coded packets makes its way closer to the destination (the others could be lost due to lossy channels). The first challenge is to decide how many

coded packets a FN should generate and send in response to receiving one coded packet. (2) Second, how should the bandwidth heterogeneity of channels from the source to different destinations be handled? (3) Third, at each hop, we have an upstream node that is sending data, and a number of downstream nodes receiving the data. The population of the downstream nodes changes over time in the sense that downstream nodes with good connectivity to its upstream sender or with more overhearing opportunities will complete receiving data (a data file for example) faster than their poorly connected counterparts. Thus, the downstream node population decreases in this case (since some nodes are not interested in the data anymore). One challenge is to enable an upstream sender to adapt its transmission data rate taking into consideration the current downstream node population.

Practical design that employs intra-flow NC for reliable multicasting in WMNs has received relatively little attention. The state-of-the-art of practical intra-flow NC-based multicast protocols such as MORE [57] and Pacifier [58] all employ the conventional end-to-end transport approach in which the source keeps on sending coded packets of a data batch. Intermediate FNs encode overheard packets and then forward the coded packets toward the multicast destination receivers. The destinations send end-to-end acknowledgements (ACKs) after they collect enough coded packets to decode the original data batch (this is similar to the TCP end-to-end ACK strategy in unicast). There is no cooperation among intermediate FNs. The number of coded packets that a FN transmits upon receiving a packet from its upstream node, so called *transmission credit (TC)*, is determined using heuristics based on measurements of the expected packet loss rates to the destinations. This approach suffers from several fundamental problems.

1. FNs may transmit coded packets much more than necessary, significantly wasting network resources and degrading the performance since it is extremely difficult to obtain an accurate estimation for the value of TC for each FN in a dynamic wireless environment.
2. While an end-to-end ACK is being propagated from a multicast receiver back to the source, the source will keep on transmitting coded packets that belong to an already delivered data batch (it stops upon receiving the ACK). These redundant packets will trigger sending yet more coded packets by the FNs, leading to bandwidth waste and more interference. This becomes even worse if the end-to-end ACK gets delayed in the cases of congestion, or lost in the situations of bad link qualities.
3. The end-to-end approach used in the existing designs cannot completely solve the unfairness issue related to heterogeneous receivers.
4. The existing end-to-end multicast protocols do not adapt the data rate at each hop to account for varying receiver population and channel conditions. Selecting of a robust low data rate to broadcast coded packets to downstream receivers does not necessarily lead to higher multicast throughput.

In this chapter, we argue that a receiver-driven hop-by-hop transport approach works better with intra-flow NC. Multiple receivers interested in a content file send requests towards the content source and a multicast tree is built. The content file is divided into batches and originally distributed from the source. An intermediate router or FN on the tree overhears coded packets sent by any neighbor (parent, grandparent, sibling, and even child nodes) and cache them in a buffer. Once collecting enough packets to decode a data batch, the FN becomes a *new source* for the cached batch. It informs its parent of

successful reception of the batch and requests the next batch. The acknowledgement and request messages are also used by FNs for flow control. A parent node would stop transmitting from a data batch after receiving acknowledges from all of its immediate children. This approach allows exploiting not only computing power (network coding) but also in-network storage (caching) at intermediate FNs, moving data as close as possible to the content requesters, i.e. the destinations.

Based on the above basic idea, we propose HopCaster, an intra-flow NC-based hop-by-hop reliable multicast protocol to achieve high-throughput over WMNs and solve the network heterogeneity issue of multicast receivers. In contrast to the existing end-to-end intra-flow NC-based multicast protocols, HopCaster completely eliminates the needs for estimating the transmission credit at FNs as well. It also simplifies multicast management and congestion control. Moreover, HopCaster employs a novel cross-layer rate adaptation mechanism to optimize the PHY data rate selection considering the receiver population changes at different stages of multicasting. We compare the hop-by-hop and end-to-end transport approaches. Our evaluations for static and dynamic scenarios show that, compared to Pacifier, a state-of-the-art intra-flow NC-based multicast protocol, HopCaster achieves up to 29% throughput improvement.

The remainder of this chapter is organized as follows. Related work is reviewed in Section 2. In Section 3, we present the HopCaster design, we also discuss the intuition underlying our design approach and the synergy between hop-by-hop transport and intra-flow NC. In Section 4, we present simulation results to compare between HopCaster with Pacifier, a state-of-the-art intra-flow NC-based multicast protocol that adopts an end-to-end transport scheme. The chapter is concluded in Section 5.

5.2. Related Work

The majority of work addressing reliability in multicast protocols over wireless networks is based on the traditional FEC, ARQ or hybrid ARQ-FEC techniques [49] [50] [51] [52] [53] where intermediate nodes or routers simply duplicate and forward packets. There are only a few practical designs and performance studies for intra-flow NC-based multicast protocols. MORE [57] is the first intra-flow NC-based protocol for reliable unicast and multicast over WMNs. In MORE, any node that overhears a transmission and that is closer to the destination than the sender may participate in forwarding the packet to the destination. This transmission participation scheme results in forming a *forwarding belt* instead of path to increase network robustness. A FN encodes packets with random network coding before forwarding them. However in MORE, multiple FNs forward the coded packets without any coordination. Belt forwarding can be inefficient, especially in multicasting, as multiple overlapped belts are formed, and the number of nodes participating in packet forwarding becomes more than necessary. Further, MORE lacks source rate control which can cause congestion.

Pacifier [58] is a state-of-the-art reliable multicast protocol for WMNs. It improves upon MORE by using a multicast tree instead of forwarding belts. Only nodes on the multicast tree perform random network coding on incoming packets. They also forward the coded packets along the tree. Pacifier also proposes a mechanism to limit the transmission rate of the source to prevent congestion using backpressure. It has been shown that Pacifier is able to achieve performance improvements over MORE.

In the designs of both MORE and Pacifier, the source computes a transmission credit (TC) for each FN through periodical packet loss rate measurements for the links in the network. The TC values are carried in the coded packets. When a FN receives a packet from its parent node, it uses the TC field to determine whether and how many coded packets to transmit to its children nodes. The successful data delivery is verified through end-to-end acknowledgements. The source continues broadcasting coded packets until it receives end-to-end ACKs from all destination receivers. Reacting to end-to-end ACKs in addition to using TCs computed at the source cause many problems, such as inaccurate estimation of TCs and inability to handle heterogeneous receivers.

Peer-to-peer (P2P) overlay networks have been widely studied [60] [61]. We consider P2P networks related to our work from an application perspective (it also aims to achieve reliable and high throughput data delivery). Further, in P2P networks, participants make a portion of their resources, such as processing power and storage available to other network participants (this is similar to FNs in our protocol where they share their caching and processing capabilities). However, in P2P systems, peers at the edge of networks form an application layer overlay with little knowledge about the underlying physical network topology. Therefore, it is difficult, if not impossible, to achieve efficient network routing. It is also difficult to take advantage of the characteristics of the underlying physical network (e.g., to take advantage of overhearing if the underlying physical network is a wireless network). Furthermore, most of P2P applications, such as BitTorrent, employ TCP for reliable transport between peers. TCP has well-known problems in wireless networks.

Research on the future Internet architecture has started recently [62] [63]. A cache-and-forward scheme has been proposed to provide unified network transport services for the future Internet [64]. However, their work mainly focuses on architectural designs and unicasting, with no NC in the picture. Therefore, our proposed protocol fits well in the cache-and-forward future architecture (due to its hop-by-hop nature) to achieve reliable and high throughput multicast applications over WMNs.

To the best of our knowledge, HopCaster is the first practical protocol that integrates NC and receiver-driven hop-to-hop transport with the optimization of link layer transmission and network layer routing for efficient reliable multicasting.

5.3. HopCaster Design

The design of HopCaster addresses the weaknesses suffered by the intra-flow NC-based multicasting protocols that are based on the end-to-end transport principle, such as Pacifier and MORE. In this section, we describe the design of HopCaster, and we compare it with that of the conventional end-to-end approach.

5.3.1. Hop-by-Hop Reliable Multicast with Network Coding

HopCaster uses a tree-based hop-by-hop opportunistic multicasting with intra-flow NC. Building the tree is initiated by requests sent from the content receivers (also called destinations) to the content source, connected by a set of intermediate wireless routers (also called forwarding nodes (FNs)). We'll describe in detail the procedure of establishing and maintaining the hop-by-hop transport tree later in this section. As shown

in Figure 5.1, the solid lines represent the parent-child links on the multicast tree, and the dashed lines indicate that a node can overhear packets from other neighbors, such as grandparents, siblings, and child nodes. We see that there can be many overhearing opportunities for a node on the multicast tree.

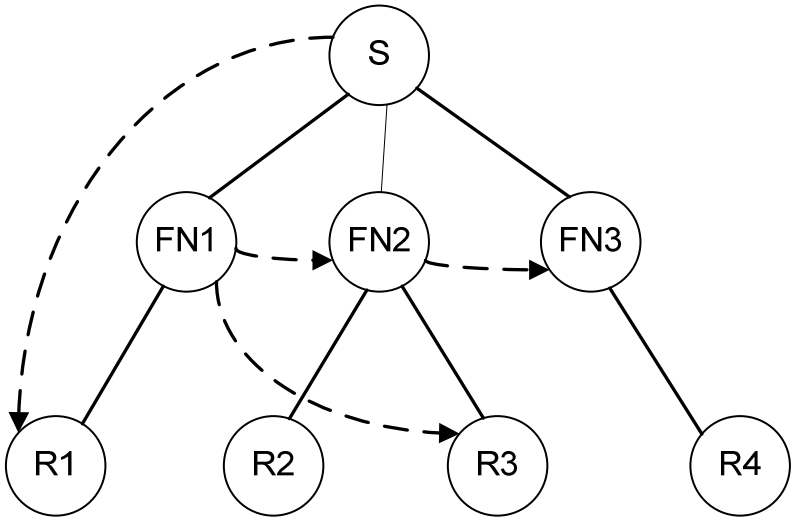


Figure 5.1: Overhearing along a multicast tree in a WMN.

A large content file can be divided into multiple batches at the source before distribution to reduce packet header overhead and decoding delay at the receiver. A batch is further divided and encapsulated into k packets. Random linear block codes [54] are applied on the packets belonging to a batch to generate coded packets. A coded packet here is a random linear combination of the k packets with coefficients chosen from a Galois field of size 2^8. The source broadcasts the coded packets when the MAC allows transmission. Each transmitted packet is also augmented with its batch ID and the vector of the random coefficients used to generate the packet.

When a FN on the multicast tree receives or overhears a coded packet from any of its neighbors, it checks whether the overheard packet is innovative, i.e., whether it is linearly independent from the packets obtained in previous transmissions (Gaussian elimination is

used for the independence check). An intermediate FN on the multicast tree caches all the innovative packets of the batch, and also performs the random linear combinations of them and sends the re-encoded packets.

Different from Pacifier and MORE, data transport in HopCaster operates in a receiver-driven hop-by-hop manner; after a FN successfully receives or overhears (from all possible neighbors) a number of innovative coded packets sufficient to decode a data batch (i.e., receives k linearly independent packets with the same batch ID), it performs the following operations:

- It sends an ACK to its parent. It indicates in the ACK message if it wants to request for next batch. A FN uses the ACK-request message for controlling the data flow (more discussion about flow and congestion control is in Section 3.3).
- The FN becomes a new source for the batch responsible for delivering this batch to its children. Note that if a batch has been requested by its children, a FN can start broadcasting random linear combinations of packets belonging to the batch without waiting to reach the full rank (i.e., k innovative packets) so as to keep the pipeline going and reduce delay. Random network coding removes transmission of duplicate copies of a packet over the wireless medium, and the probability that a node receive non-innovative packets from its neighbors exponentially decreases with the code length [6].

The parent starts sending the next batch as soon as it receives the first acknowledgement for the current batch and a request for the next batch from one of its children. The parent node sends the batches in a round robin fashion; i.e., after the last batch is acknowledged by a child, it goes back to send from the batches that haven't been

acknowledged by all its children yet. This continues until all batches are acknowledged by all children. With this scheduling scheme, the children with good link quality can finish receiving the whole file faster without waiting for its poorly connected counterparts. Children with poor connection would be served in later rounds.

A FN keeps the received batches of data in its cache (buffer memory or storage) as long as it is still in the multicast group and has the capacity available in its buffer. FNs use the Least Recently Used (LRU) policy for its buffer replacement. The cached copies may be used during network topology changes, or by receivers newly joining the multicast group as described below.

The source and FNs on the multicast tree maintain a soft-state *batch request and ACK table* (BRAT). This table keeps track of batches with pending requests (IDs of batches that were requested but not served yet). It also keeps track of the acknowledgement state of batches (for each batch ID, the IDs of child nodes that acknowledged the batch). Every parent node knows the list of its children through multicast tree construction.

HopCaster uses a subscribe/join scheme to build the multicast tree and to populate and refresh the (BRAT). The tree roots at the source and consists of all the shortest paths from the source to the destination receivers. It is assumed that a unicast routing protocol such as OLSR [65] or HWMP [66] is used to construct unicast paths in the network employing the expected transmission count (ETX) metric [67]. Our HopCaster protocol can be essentially implemented as a shim layer on top of layer 3. A destination receiver interested in the content file sends a *request* message to its parent on the path toward the source. The request message contains a *batch ACK field* (BAF), which is a bitmap indicating which batches have been successfully received by the sending node. The

request message is propagated and processed hop-by-hop along the path toward the source. If the parent node has cached the batches needed by the sender of the request message (the corresponding bits in the BAF bitmap has not been set yet), the needed batches get served from this parent node to save bandwidth and reduce delay. The parent node updates the BAF field in the request message by setting the bit corresponding to the batch that it would serve. It then sends the request message to its parent. If at some point all the BAF bits get set, then the request message is not forward upstream anymore. Otherwise, the request message propagates until it gets to the root of the tree.

In HopCaster, BRAT entries regarding received request messages are maintained as a soft state that is discarded after a timeout. A destination receiver is responsible for sending its request periodically toward the source in order to refresh the state in the BRAT and to indicate that its request is still valid. For bandwidth efficiency, request messages can be combined with batch ACK messages if possible.

When changes in network topology are detected or when links break, the path between the destination and the source is reconstructed by the underlying unicast routing protocol. To speed up the recovery process, the affected nodes are informed of the events of link breaks and routing changes. If a node changes its parent node towards the source, the node sends a request message to its new parent node on the reconstructed path toward the source. The request message contains the BAF field to indicate its progress in content delivery, i.e., which batches it has successfully received. If a node loses a child node due to routing changes, it removes the state for that child. Note that in HopCaster, a new receiver can be efficiently served by the closest node that has the requested content locally cached.

5.3.2. Multicast Rate Adaptation

The hop-by-hop nature of HopCaster, in which a FN keeps track of its children and serves as a new source for locally cached content, motivates a cross-layer rate adaptation algorithm to assist HopCaster in determining optimal transmission rate at the PHY layer. Existing radios such as IEEE 802.11 [68] support multi-rate capability at physical layer by using different modulation and channel coding (MCS) modes. However there is no link-layer feedback for multicast packets in 802.11. While most rate adaptation mechanisms try to maximize the PHY rate for unicast, a low basic rate is simply selected for multicast, which is certainly not optimized.

Along the multicast tree, the source node or a FN have multiple children. The children with good channel conditions and more overhearing opportunities will finish receiving a data batch earlier. Therefore, for each parent node, the population of children nodes will change while sending from a batch. In HopCaster, a FN on the multicast tree periodically reports its link conditions to its parent. The reporting messages can be combined with the batch request and ACK messages if possible to reduce the overhead. The parent then knows which children still need the coded packets for a particular batch (based on the BRAT) as well as their channel conditions. It selects a PHY rate for a packet transmission to maximize the throughput at its intended children.

Given a radio technology and PHY mode, the packet loss probability can be estimated from channel SNR γ. Just as an example, DPSK, QPSK and CCK modulations with different channel coding schemes are used in IEEE 802.11b PHY modes $m = 1, 2, 3$ and 4 to get transmission rates of 1 Mbps, 2 Mbps, 5.5 Mbps, and 11 Mbps, respectively. The packet loss probability can be calculated by [69]:

$$P_e(L,\gamma,m) = 1 - \left\lfloor 1 - P_e^1(24,\gamma) \right\rfloor \times \left\lfloor 1 - P_e^m(L,\gamma) \right\rfloor \quad 5.1$$

where $P_e^1(24,\gamma)$ is the error probability of the physical layer convergence procedure (PLCP) sublayer header that is 24 bytes long and is always transmitted with PHY mode 1. $P_e^m(L,\gamma)$ is the error probability for the *L*-byte data payload and MAC header transmitted with PHY mode *m* that can be expressed as

$$P_e^m(L,\gamma) = 1 - (1 - P_b^m(\gamma))^{8L} \quad 5.2$$

where $P_b^m(\gamma)$ is the bit error rate (BER) that corresponds to SNR γ under PHY mode *m*.

From Equation 5.1, it is clear that the packet error probability depends on the PHY mode, the packet size, and the channel SNR γ. Note that this is true for other radios such as 802.11a/g/n even they use more complex PHY modulation, channel coding, and antenna techniques in different PHY modes. In general, a relationship database between the packet loss rate and the used PHY mode under different packet sizes and channel SNR values can be built through modeling or offline measurements.

Different multicast receiving nodes have different channel SNR values. Given the received channel SNR γ_t for a receiving node *t*, the $P_e^t(L,\gamma_t,m)$ can be derived for each PHY mode *m* based on the relationship database. Suppose a multicast transmitter *N* currently has *T* receiving child nodes, the objective is to find the best PHY mode for the multicast transmitter to achieve the overall maximum throughput among all its intended receiving nodes. The objective function can be formulated as

$$\max \sum_{t=1}^{T} r(m)(1 - P_e^t(L,\gamma_t,m)) \quad 5.3$$

where $r(m)$ is the data rate for the PHY mode m used by the multicast transmitter.

Each child node on the multicast tree measures its channel SNR for the packets received from its parent node and periodically feeds the channel SNR measurements back to its parent node. The parent node can then estimate the objective function and select the best PHY rate.

Once the parent node receives an ACK for a batch from one of its children, the PHY rate adaptation mechanism is informed of this event, and it readjusts its multicast PHY rate to obtain the maximum total throughput based on the link quality of the children that have not completed reception of this batch.

5.3.3. Comparison: Hop-by-Hop vs. End-to-End

As discussed before, the hop-by-hop transport avoids the need to estimate the value of TC that a FN uses to determine how many coded packets it should send. Furthermore, we argue that the hop-by-hop transport approach in HopCaster offers several other advantages over the end-to-end transport used in Pacifier and MORE.

- The hop-by-hop transport handles the heterogeneous receivers better. Consider a situation where a bottleneck link is close to a destination node. If the end-to-end transport protocol is used, the source will keep transmitting the packets until an end-to-end ACK from the worst receiver is received. Alternatively, hop-by-hop transport avoids this deficiency because the FN at the upstream of the bottleneck link will cache the data and inform its parent node to stop sending unnecessary packets. The data is pushed towards the destinations as close as possible and network bandwidth is utilized more efficiently. The performance of multicast is not limited by the path of

the worst connected receiver. A lossy and low-capacity wireless link only affects the receivers behind it on the multicast tree.

- The hop-by-hop transport greatly simplifies flow and congestion control. With the end-to-end approach, there are points in the network between the source and destination where congestion may occur from traffic aggregation. Certain congestion control mechanism that dynamically adjusts the source transmission rate to keep the aggregate traffic volume below the network congestion level is required in order to maintain flow balance, otherwise packet loss would occur. In HopCaster, by contrast, all communications are local and flow balance is maintained at each hop through request (pull) and ACK messages. A child node sends a request to its parent only if it is *ready* to accept a new batch. There is no need for additional techniques to control congestion in the middle of a path. This is not the same as backpressure-based hop-by-hop flow control [70], where backpressure between adjacent nodes is used for a node to adjust its packet forwarding rate of a continuous flow. Note that a FN in HopCaster does not have FIFO queues but a LRU memory that decouples the hop-by-hop control loops, compensates bandwidth fluctuations, and reduces the impact of multicast receiver heterogeneity. In case all the data batches in the buffer are in current use due to very small storage size, a node can simply delay sending its request to the parent node until the buffer space is available.
- The advantages of hop-by-hop transport are even more pronounced when the network becomes congested or nodes are mobile. Even for a single destination, the end-to-end connection over a path of n hops would break or deteriorate as soon as any link in the path failed or degraded, triggering route repair or route re-discovery mechanisms. The

probability of disconnection or degradation on the *n* hop path would be roughly *n* times that of a single hop. The situation is worsened in multicast since more receivers and FNs are involved. Thus, a hop-by-hop transfer is far more reliable and allows delivery from intermediate nodes after route repair.

- Propagation time of a 1-hop ACK is shorter than that of a multi-hop end-to-end ACK. Therefore the number of redundant packets injected into the network while a 1-hop ACK is on its way back to a parent node is much smaller than that when an end-to-end ACK is sent to the source node from the destination. Moreover, the 1-hop ACK is much less possible to be delayed or lost due to congestion and transmission errors than the end-to-end ACK, In HopCaster, transmission of coded packets at a FN is not triggered by receiving a packet from its parent, rather than the requests from the children. Thus even if the 1-hop ACKs get lost or delayed, a packet originated from a parent node will not trigger more transmissions from the child nodes (as the case in MORE and Pacifier).
- The hop-by-hop feedback control allows local optimization along the multicast tree. One such optimization proposed by HopCaster is to enable each parent node to adapt its radio transmission rate to maximize multicast throughput by exploiting the children's population changes and wireless channel variations.

5.4. Performance Evaluation

To evaluate the performance of HopCaster, we use Network Simulator 2 (NS-2) [71], a widely used simulator in networking research community, to simulate both HopCaster and Pacifier and compare their performance under various settings. We choose Pacifier,

not MORE, for performance comparison with HopCaster because it is reported in [58] that Pacifier achieves better performance than MORE.

We performed our evaluation under many different network topologies and parameter settings. We report results with the following settings. We have a total of 50 nodes. We generate 10 scenarios, each corresponding to a topology by placing 50 nodes randomly in a 1000×1000 square meter area. There is one content source node and 9 destination receivers requesting for the content file. The source and destinations are chosen randomly in each scenario. Note that a node could be selected as a destination and a FN at the same time. Further, we simulate a realistic wireless channel with a shadowing propagation model [72] and we assume the maximum radio transmission range of 276 meters. In our simulations we implemented the radio multi-rate capability. as described later in this section.

In the simulations, the size of the file to be transmitted from the source to the multicast destinations is 2.14 MB. The source divides the file into 48 batches, each of which is divided into 32 packets with 1460 bytes payload in a packet. To collect the packet loss probability values required by Pacifier to compute the TCs, following Pacifier implementation [58], we allow the 50 nodes involved in the simulation to periodically exchange hello messages. By keeping track of the lost hello messages, the loss probability values of all the links in the network are calculated. Next we provide our simulation results for the scenarios with static and dynamic multicast trees, and explain the gains attained by HopCaster.

5.4.1. Static Multicast Experiments

In those experiments, the multicast tree is established at the beginning of the experiment, and no new receivers are joining the tree to request the content file during the course of data transmission. The dynamic case will be investigated in the next section. Figure 5.2 shows the average throughput of Pacifier and HopCaster in the 10 different scenarios, in addition to the overall average throughput over all the scenarios. Here the throughput of a destination receiver under one scenario is calculated by dividing the file size by the time that the destination receives the whole file. Then averaging the throughput values over all destination receivers in a scenario, we obtain the average throughput for that scenario. The transmission data rate in those experiments is set to 2Mbps.

As shown in Figure 5.2, HopCaster achieves a higher throughput compared to Pacifier in all scenarios. The gain ranges from 6% (scenario 3) to 29% (scenario 9), and the average is 12%. The higher throughput of HopCaster results from injecting fewer intra-flow NC-coded packets into the network, reducing the number of required transmissions. This can be further clarified by Figure 5.3. The figure shows the number of transmissions by HopCaster and Pacifier in all the scenarios. Each bar in the graph is divided into two parts: the bottom part corresponds to the number of coded data packet transmissions and the top part corresponds to the number of ACK packet transmissions. Note that in HopCaster, the ACKs are hop-by-hop, and in Pacifier the ACKs are end-to-end. HopCaster results in sending fewer *data packets* and more *ACK packets* than Pacifier. The hop-by-hop ACK in HopCaster significantly reduces the number of redundant data packets injected into the network. Thus, HopCaster has less total number of packet transmissions (data and ACK) in all scenarios except scenario 4 (both protocols

have almost the same number of transmissions). In addition, the channel time of sending a data packet is much larger than that of an ACK packet because the ACK packet is much smaller. Figure 5.4 shows the redundancy in both protocols. Redundancy is computed as the total number of bytes in all transmitted packets (data and ACK) divided by the file size in bytes. Note that HopCaster results in much less redundancy over all scenarios.

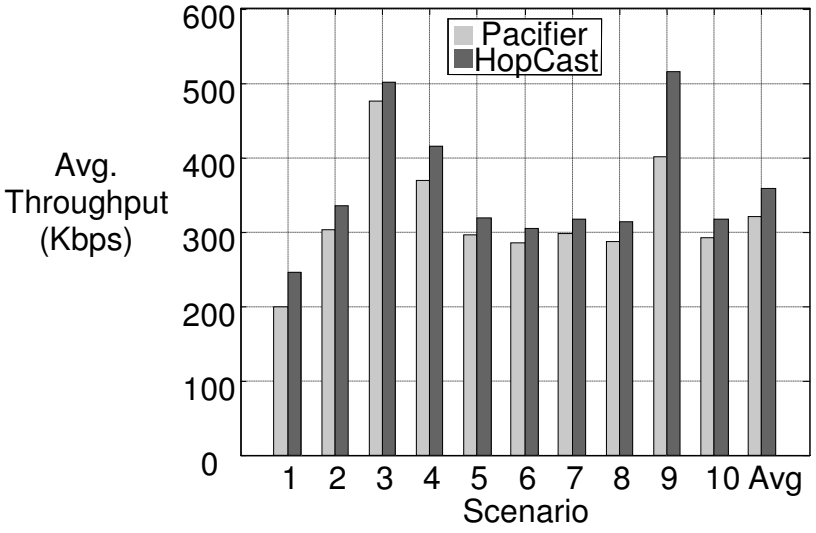


Figure 5.2: Average throughput of HopCaster and Pacifier under 10 different scenarios.

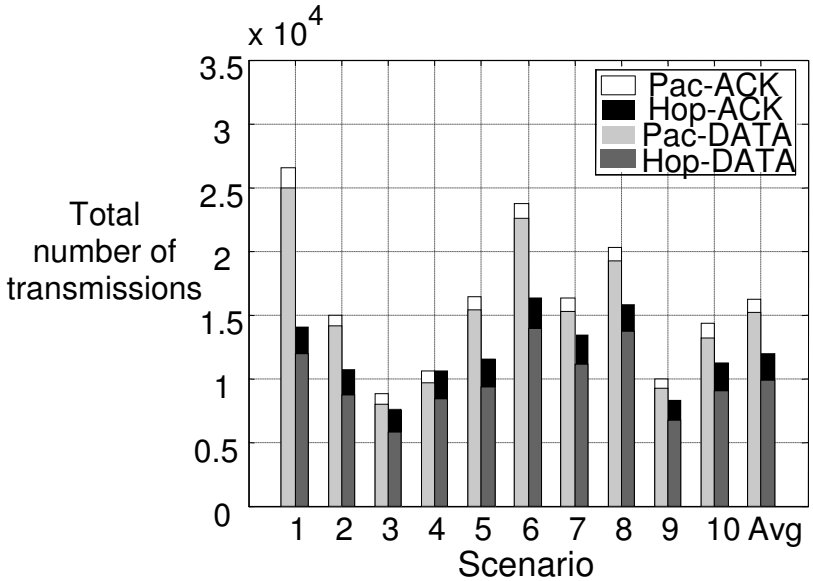


Figure 5.3: The number of transmissions (ACK and data packets) of HopCaster and Pacifier under 10 different scenarios.

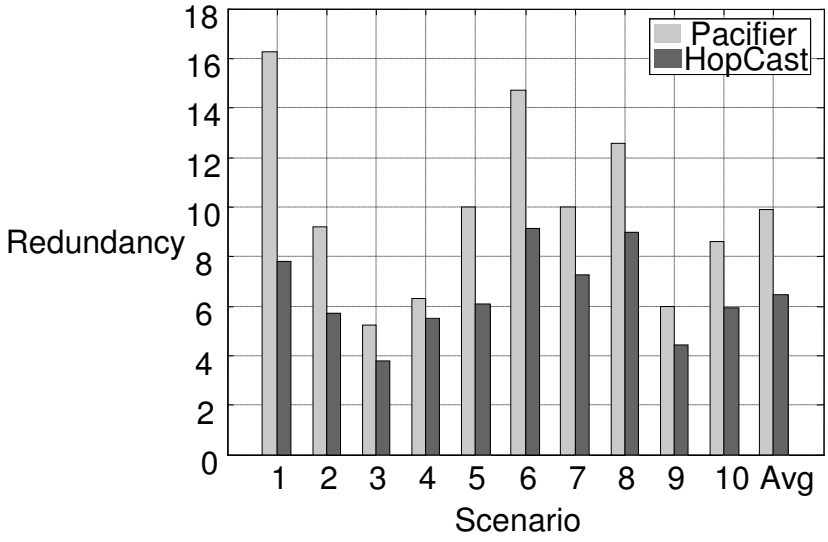


Figure 5.4: The number of redundant bytes that are injected into the network by HopCaster and Pacifier in 10 different scenarios.

5.4.2. Dynamic Multicast Experiments

As opposed to the previous static multicast simulation experiments where no new receivers were joining the multicast tree to request the content file during transmission, this section addresses the dynamic multicast case, in which new receivers request to join the tree for receiving the content data during transmission. In particular, the simulation starts with one source and 8 receivers, i.e., one of the 9 receivers in previous simulation experiments will be idle at the beginning of the simulation experiment. At some point of time, that node will request joining the multicast tree. For HopCaster, the joining node will use the request mechanism described in Section 5.3.1. For Pacifier, as described in [58], the joining node will send a join message toward the source (the root of the tree) which in turn will register the joining node as a new multicast receiver. We compare the new receiver's throughput under HopCaster and Pacifier in 10 different scenarios. In each

scenario, the new receiver and the time of the request will be chosen randomly (the same randomly chosen values will be used for both HopCaster and Pacifier in an experiment). Figure 5.5 shows that the new receiver's throughput under HopCaster is much greater than that with Pacifier. The reason is that in HopCaster, the new receiver can be immediately served from the closest upstream node in the multicast tree, where in Pacifier, only the source can serve a new node since the source needs to compute the TCs for the FNs to send the coded packets to the newcomer. This demonstrates a desirable feature of the hop-by-hop transport approach, where over time, the content will move deeper in the multicast tree and closer to the destinations. A new receiver will be served by the closest node that has cached the requested content.

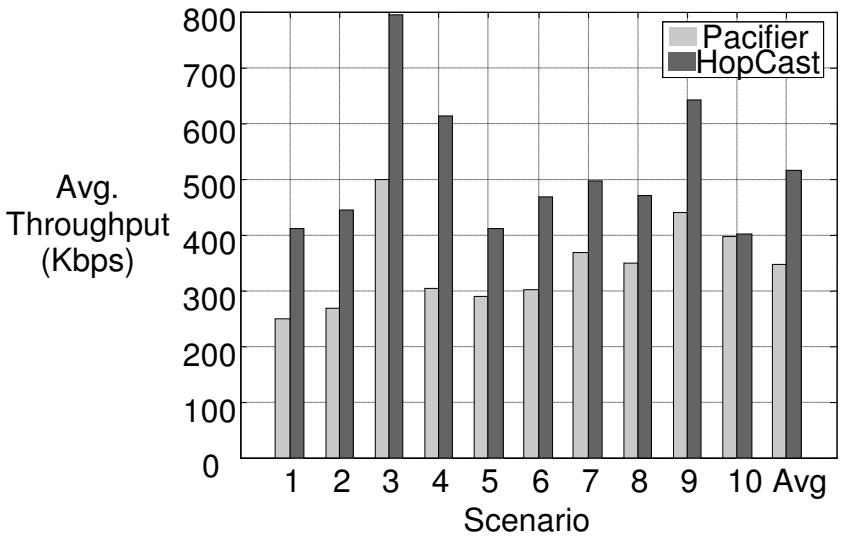


Figure 5.5: Average throughput of the new receiver that dynamically joins the multicast group with HopCaster and Pacifier in 10 different scenarios.

5.4.3. Multicast Rate Adaptation

In this section we demonstrate the throughput gain of the rate adaptation algorithm described in Section 5.3.2. In our simulations, children nodes report the SNR of received

packets to their parent node every 10 seconds. Whenever a parent node is about to transmit a packet, it evaluates the optimal data rate according to Equation 5.3 for m=1, 2, 3, 4 that correspond to data rates of 1 Mbps, 2 Mbps, 5.5 Mbps, and 11 Mbps, respectively. Equation 5.3 is evaluated only for children nodes that have not acknowledged the current batch. For comparison, Figure 5.6 illustrates the average throughput values for the 10 scenarios using the adaptive data rates as well as 2 fixed data rates: 2Mbps and 11Mbps.

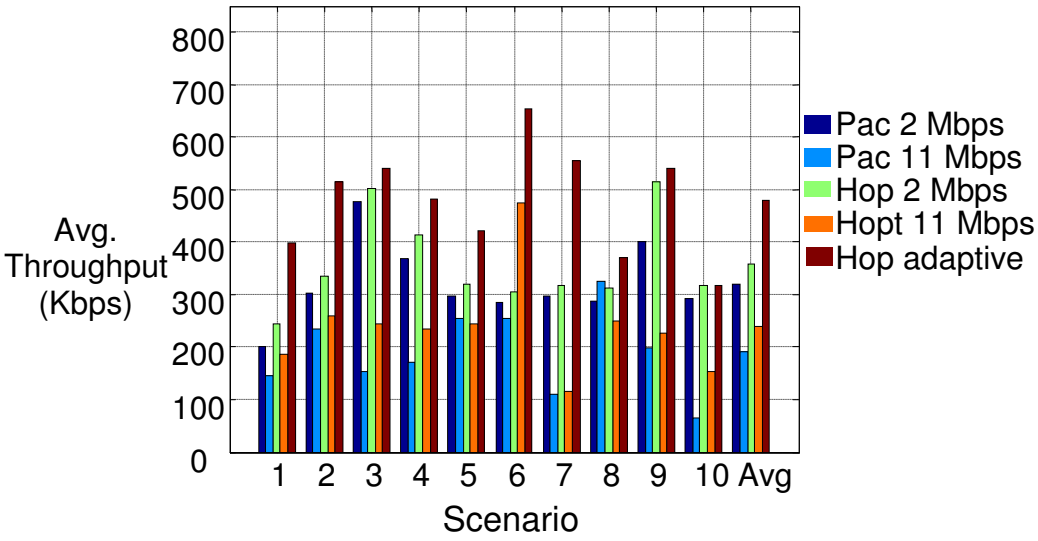


Figure 5.6: Average throughput of HopCaster and Pacifier under 10 different scenarios using 2Mbps, 11Mbps, and adaptive data rates.

The performance of both HopCaster and Pacifier degrades under 11 Mbps compared to 2 Mbps. This is because of high packet loss rate at 11 Mbps, which incurs more delay in sending data to destinations. Furthermore, the ACK messages may be lost, leading to injecting more redundant packets into the network before switching to next non-acknowledged batches. Note that the cross-layer rate adaptation mechanism greatly improves the throughput in all the experiment scenarios. The reason is that the rate adaptation algorithm enables a parent node to estimate the current conditions of the

wireless links to its targeted children. Therefore it is able to dynamically select a data transmission rate to maximize the overall throughput across all its targeted children.

5.5. Discussion

To design a practical protocol that delivers the benefits of intra-flow NC in improving multicast performance over WMNs, many challenges need to be addressed, such as how many coded packets a FN should send and how to handle the heterogeneity of multicast receivers. In this chapter, we have designed HopCaster, a novel reliable multicast protocol that incorporates intra-flow NC with receiver-driven, hop-by-hop transport. Compared to the existing intra-flow NC-based multicast protocols that all follows the conventional end-to-end transport principle, HopCaster design eliminates the need for estimating the number of coded packets each FN should transmit, and simplifies multicast management and congestion control. We also proposed a cross-layer rate adaptation mechanism that enables HopCaster to optimize PHY rate selection in hop-by-hop multicast by exploiting the changing receiver population and wireless channel variations. Our simulations show that compared to Pacifier, a state-of-the-art intra-flow NC-based multicast protocol, HopCaster greatly reduces the number of required transmissions over wireless medium to deliver multicast data and achieves up to 29% throughput improvement. Furthermore, we showed that the advantages of HopCaster are more prominent in the situations that new nodes dynamically request for the contents.

Chapter 6

Conclusion and Future Work

In this chapter we provide a conclusion and outline directions for future work.

6.1. Conclusion

In this dissertation we studied the practical aspects that arise when deploying DVC over real visual sensors. Further, we proposed and evaluated HopCaster; a scheme that employs the cache-and-forward transport strategy to achieve high throughput transmission in multicasting over wireless mesh networks while maintaining 100% packet delivery ratio.

In Chapter 1 we provided a brief background about Distributed Source Coding (DSC) and Network Coding (NC).

In Chapter 2 we studied DVC when deployed over real visual sensor. We described the design and implementation of a DVC codec that we deployed over the popular Micaz/Cyclops and Imote2/IMB400 visual platforms. Further, we reported a summary of accurate measurements regarding the distribution of energy consumption over the

different DVC operations. Moreover, we carried out an operational energy-distortion analysis for a variety of options available to RDVC on visual sensors.

In Chapter 3 we presented our approach towards a rate-adaptive DVC solution that eliminates the need for feedback messages from the base-station to the visual sensors; this solution makes utility of the correlation between the source rate and the conditional entropy of the WZ and key frames.

In Chapter 4 we employed and evaluated the performance of polar codes in DVC. Further, we compared the performance of polar codes with the more established and more investigated Low Density Parity Check Accumulate (LDPCA) codes in the context of DVC. We showed that polar codes offer a clear advantage when coding smaller size image blocks. Consequently, polar codes could represent a viable solution for distributed sensor networks that capture a low-resolution video signal at each visual sensor node.

In Chapter 5 we proposed HopCaster, a novel protocol that incorporates intra-flow NC with hop-by-hop transport to achieve high-throughput reliable multicast over wireless mesh networks. We showed that by adopting the hop-by-hop transport strategy, HopCaster avoids many problems incurred by the traditional end-to-end transport strategy.

6.2. Future Work

The intended future work falls into the following directions:

- An essential component of traditional video compression standards, such as H.264, is motion estimation (ME) that requires intensive computations that are mainly related to block-wise motion search. For DVC to compete with standard

video compression schemes, there is a need to take advantage of ME at the decoder side. As we saw in Section 2.3 (Figure 2.2), there has been attempts to utilize ME in the context of DVC, mainly by allowing the encoder to provide the decoder with information that assist it in performing ME. This information can take the form of hash or cyclic redundancy check (CRC) data. Part of our future plan is look deeper into this aspect, and to find ways that assist the decoder to perform ME using minimal amount of overhead data.

- Part of our future plan is to bring NC and DVC together to realize a practical and efficient data networking platform that fits in resource constrained networks. This is accomplished by integrating NC and DVC so that the two schemes work collaboratively. This is important in situations where the collected data demonstrates a degree of correlation (DVC proves to be efficient in compressing data before transmission), and in the meanwhile the collected data is propagated through a wireless network where overhearing is possible (NC is efficient in eliminating unnecessary transmissions and hence maintain high throughput).
- In this dissertation the focus have been on DVC over one camera, i.e., the case where DVC aims to compress temporally consecutive frames that are captured by one camera. In fact, DVC can utilize the correlation amongst frames captured by many cameras, since when the cameras are located in spatial vicinity, the captured frames are expected to demonstrate mutual redundancy across the different cameras. Part of our future plan is to explore this aspect and to devise algorithms that can efficiently quantify the amount of disparity between frames

captured across multiple cameras and utilize the correlation to achieve maximum compression.

- We showed in chapter 4 that polar codes offer a clear advantage when coding smaller size image blocks. Our results were in terms of compression performance. Although this result encourages the use of polar codes, it needs more evidence. As we saw in Chapter 3, a more conclusive measure would be in terms of the overall power consumption that takes into account savings in transmission energy in addition to computational energy. As we mentioned earlier, although polar codes have low encoding complexity, they are more computational intensive compared with LDPC codes (and $O(N \log N)$ for polar codes vs. $O(N)$ for LDPC codes). Part of our future plan is to investigate the advantage of using polar codes in terms of the overall power consumption (computation plus transmission).
- Results in Chapter 4 show that polar codes offer a clear advantage when coding smaller size image blocks compared to LDPCA codes. The results in Chapter 4 were attained when using the successive cancellation (SC) algorithm to decode polar codes. It is reported in the literature that the belief propagation (BP) algorithm can be used to decode polar codes, and it achieves higher performance compared to SC. Part of our future plan is to incorporate BP in our DVC codec and compare it with LDPCA.

APPENDIX A: LDPCA Codes

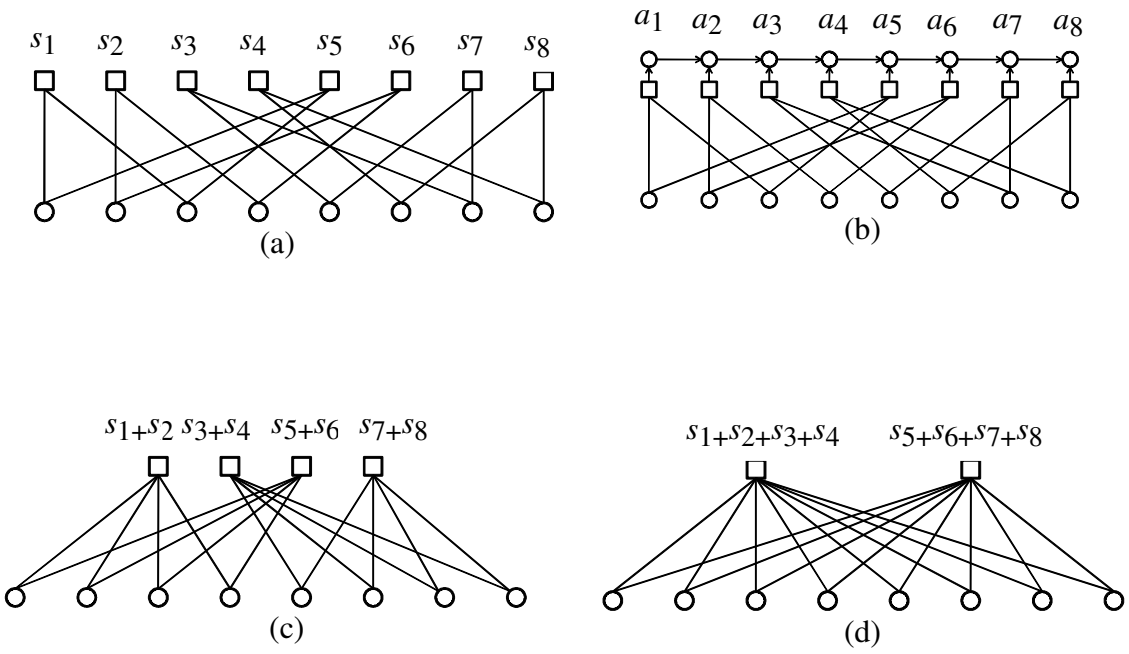


Figure A.7.1: LDPCA code with different rates.

Figure A.7.1 demonstrates the idea of LDPCA codes. We have an 8×8 parity check matrix that is represented by the Tanner graph [3] in Figure A.7.1 (a). When this graph is used to compress a WZ bitstream of 8 bits, a syndrome of 8 bits will be generated. Note that no compression is achieved in this case. LDPCA [36] suggests achieving *different levels* of compression by accumulating (summation modulo 2; equivalent to binary XOR) the syndrome bits:

$$a_1 = s_1 \qquad\qquad a_i = a_{i-1} + s_i \text{ , where } i = 2 \text{ to } n$$

The accumulation process is shown in Figure A.7.1 (b). Compression is achieved by transmitting only a subset of the accumulated syndrome bits. For example, if we transmit

a_2, a_4, a_6, a_8, then a compression ratio of 1/2 is achieved. At the decoder, four syndrome bits are generated as follows:

$$s_1 + s_2 = a_2 \qquad s_5 + s_6 = a_4 + a_6$$

$$s_3 + s_4 = a_2 + a_4 \qquad s_7 + s_8 = a_6 + a_8$$

The four syndrome bits are used for decoding over a new Tanner graph. This graph (Figure A.7.1 (c)) results from merging each pair of consecutive check nodes in the original Tanner graph. To achieve a compression ratio of 1/4, only a_4 and a_8 are transmitted. A new Tanner graph (Figure A.7.1 (d)) is constructed by merging each set of consecutive 4 check nodes in the original graph. Further, two syndrome bits are generated:

$$s_1 + s_2 + s_3 + s_4 = a_4 \qquad s_5 + s_6 + s_7 + s_8 = a_4 + a_8$$

REFERENCES

REFERENCES

[1] J. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transaction on Information Theory*, vol. IT-19, pp. 71–480, July 1973.

[2] S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): Design and construction," in *Proc. IEEE Data Compression Conference*, , Snowbird, UT, Mar. 1999.

[3] W.E Ryan, "An Introduction to LDPC Codes," in *CRC Handbook for Coding and Signal Processing for Recording Systems (V. Vasic, ed.)*.: CRC Press, 2004.

[4] N. Cai, S.-Y. Li, and R. Yeung R. Ahlswede, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, pp. 1204–1216, July, 2000.

[5] D. Fulkerson L. Ford, "Maximal flow through a network," *Canadian Journal of Mathematics*, pp. 399–404, 1956.

[6] R. W. Yeung, and N. Cai S.-Y. R. Li, "Linear network coding," *IEEE Transaction on Information Theory*, Feb. 2003.

[7] E. Arikan, "Channel polarization: A method for constructing capacity achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, pp. 3051–3073, 2009.

[8] A. Aaron, S. Rane, and D. Rebollo-Monedero B. Girod, "Distributed Video Coding," *IEEE Special Issue on Video Coding and Delivery*, vol. 39, no. 1, pp. 71-83, Jan 2005.

[9] A. D. Wyner and J. Ziv, "The rate-distortion function for source coding with side informationat the decoder," *IEEE Transaction on Information Theory*, vol. IT-22, pp. 1-10, Jan. 1976.

[10] S. Karande, and H. Radha K. Misra, "Multi-Hypothesis Based Distributed Video Coding using LDPC Codes," in *Proc. Allerton Conference on Communication, Control, and Computing*, Allerton, IL, Sept.2005.

[11] I. H. Tseng and A. Ortega, "Rate-distortion Analysis and Bit Allocation Strategy for Motion Estimation at the Decodec Using Maximum Likelihood Technique in Distributed Video Coding," in *Proc. IEEE International Conference on Image Processing (ICIP07)*, TX, Sep., 2007.

[12] R. Puri and K. Ramchandran, "PRISM: A New Robust Video Coding Architecture Based on Distributed Compression Principles," in *Proc. Allerton Conference on Communication, Control, and Computing*, Allerton, IL, Oct. 2002.

[13] B. Code, E. Kaiser, M. Shea, and L. Bavoil W. Feng, "Panoptes: scalable low-power video sensor networking technologies," in *Proc. of ACM Multimedia*, CA, USA, Nov. 2003.

[14] D. Ganesan, P. Shenoy, and Q. Lu P. Kulkarni, "SensEye: a multi-tier camera sensor network," in *Proc. of ACM Multimedia*, 2005.

[15] R.Halloush, K. Misra and H. Radha, "Practical Distributed Video Coding over Visual Sensors," in *Proc. Picture Coding Symposium (PCS09)*, Chicago, IL, USA, May 6, 2009.

[16] R. Halloush and H. Radha, "Practical Distributed Video Coding Based on Source Rate Estimation," in *Proc. 44th Conference on Information Sciences and Systems (CISS'10)*, Princeton University, NJ, USA, March 2010.

[17] R. Baer, O. Iroezi, J. Garcia, J. Warrior, D. Estrin, M. Srivastava M. Rahimi, "Cyclops: in situ image sensing and interpretation in wireless sensor networks," in *Proc. ACM Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, CA, Nov. 2.

[18] Crossbow Corporation. [Online]. http://www.xbow.com

[19] Atmel Corporation. [Online]. http://www.atmel.com/products/avr

[20] ChipCon Inc. [Online]. http://www.chipcon.com

[21] ZigBee Alliance Homepage. [Online]. http://www.zigbee.org

[22] A. Goode, D. Goel, and I. Nourbakhsh A. Rowe, "CMUcam3: an open programmable embedded vision sensor," Carnegie Mellon Robotics Institute, Technical Report 2007.

[23] C. Park and P. Chou, "eCAM: ultra compact, high data-rate wireless sensor node with a miniature camera," in *Proc. International Conference on Embedded Networked Sensor Systems (SenSys06)*, New York, NY, USA, 2006.

[24] A. Abbo, B. Schueler, A. Danilin R. Kleihorst, "Camera mote with a high-performance parallel processor for real-time frame-based video processing," in Proc of IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS07)," , London, UK, 5 Sept. 2007.

[25] Marvell Corporation. [Online]. http://www.marvell.com

[26] T. Melodia and K. Chowdhury I. Akyildiz, "Wireless Multimedia Sensor Networks: Applications and Testbeds," *Proceedings of the IEEE Inst*, vol. 96, pp. 1588–1605, 2008.

[27] P. Levis, D. Culler, E. Brewer D. Gay, *nesC 1.1 Language Reference Manual*., May 2003.

[28] Tinyos: An operating system for networked sensors. [Online]. http://www.tinyos.net

[29] R. Zhang, and B. Girod A. Aaron, "Wyner-Ziv coding of motion video," in *Proc. Asilomar Conference on Signals and Systems*, Pacific Grove, CA, Nov. 2002.

[30] E. Setton, and B. Girod A. Aaron, "Towards practical Wyner-Ziv coding of video," in *Proc. IEEE International Conference on Image Processing*, Barcelona, Spain, Sept. 2003.

[31] S. Rane, E. Setton, and B. Girod A. Aaron, "Transform domain Wyner-Ziv codec for video," in *Proc. SPIE Visual Communications and Image Processing*, San Jose, CA, Jan. 2004.

[32] S. Rane and B. Girod Aaron, ""“Wyner-Ziv video coding with hash-based motion-compensation at the receiver," in *Proc. IEEE International Conference on Image Processing*, Singapore, Oct. 2004.

[33] R. Szewczyk and D. Culler J. Polastre, "Telos: Enabling Ultra-Low Power Wireless Research," in *Proc. the fourth International Conference on Information Processing in Sensor Networks (IPSN05)*, 2005.

[34] K. Ozonat, "Lossless Distributed Source Coding for Highly Correlated Still Images," Dept. of Electrical Engineering, Stanford University, Stanford, CA., Technical report 2000.

[35] "H.264/MPEG-4 Part 10: Transform and Quantization," in *White paper at http://www.vcodex.com*.

[36] A. Aaron, B. Girod D. Varodayan, "Rate-Adaptive Codes for Distributed Source Coding," in *Proc. Asilomar Conference on Signals, Systems, and Compuers,* Pacific

Grove, CA, 2005.

[37] Agilent Infiniium Oscilloscope. [Online]. http://www.agilent.com.

[38] Analog Devices AD620 Instrumentation Amplifier. [Online]. http://www.analog.com

[39] M. Krämer and A. Geraldy, "Energy measurements for micaz node," University of Kaiserslautern, Kaiserslautern, Germany, Technical Report 2006.

[40] J. Thomas T. Cover, *Elements of Information Theory*.: Wiley Series in Telecommunications, 1991.

[41] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, Second Edition. ed.

[42] *Intel Wireless MMX Technology Developer Guide*., August, 2002.

[43] A. Majumbar, P. Ishwar, and K. Ramchandran R. Puri, "Distributed video coding in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, pp. 94 – 106, July 2006.

[44] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proc. IEEE DCC*, 2002.

[45] S. B. Korada, *Polar codes for channel and source coding*. Ph.D. dissertation, EPFL, Lausanne, Switzerland, July 2009.

[46] L. Rad, H. Aghajan I. Downes, "Development of a Mote forWireless Image Sensor Networks," in *Proc. of Cognitive Systems and Interactive Sensors (COGIS)*, Paris, France, March 2006.

[47] D. Prashanth, S. Fong, and H. Aghajan S. Hengstler, "Mesh-Eye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance," in *Proc. of the 6th International Symposium on Information Processing in Sensor Networks (IPSN07)*, April 2007.

[48] H. Cronie and S. B. Korada, "Lossless Source Coding with Polar Codes," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Texas, USA, June 2010.

[49] K. Obraczka, S.-J. Lee, and M. Gerla K. Tang, "A reliable, congestion controlled multicast transport protocol in multimedia multi-hop networks," in *Proc. of WPMC*, 2004.

[50] L. Rizzo and L. Visicano, "RMDP: an FEC-based reliable multicast protocol for wireless environments," *Mobile Computing and Communications Review*, vol. 2, 1998.

[51] Y. Yi, K. Obraczka, S.-J. Lee, K. Tang, and M. Gerla V. Rajendran, "Combining source- and localized recovery to achieve reliable multicast in multi-hop ad hoc networks," in *Proc. of IFIP Networking*, 2004.

[52] E. Pagani and G. Rossi, "Reliable broadcast in mobile multihop packet networks," in *Proc. of ACM MOBICOM*, 1997.

[53] H. Baraka, and A. Fahmy A. Sobeih, "ReMhoc: A reliable multicast protocol for wireless mobile multihop ad hoc networks," in *Proc. of IEEE CCNC*, 2004.

[54] M. M´edard, J. Shi, M. Effros, and D. Karger T. Ho, "On randomized network coding," in *Proc. of Allerton*, 2003.

[55] H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft S. Katti, "Xors in the air: practical wireless network coding," in *Proc. of ACM SIGCOMM 2006*, Sept. 2006.

[56] J. Zhang and Q. Zhang, "Cooperative Network Coding-Aware Routing for Multi-Rate Wireless Networks," in *Proc. of IEEE Infocom*, 2009.

[57] M. Jennings, S. Katti, and D. Katabi S. Chachulski, "Trading structure for randomness in wireless opportunistic routing," in *ACM SIGCOMM*, 2007.

[58] Y. Liu, C. Wang D. Koutsonikolas, "Pacifier: high-through, reliable multicast without crying babies in wireless mesh networks," in *INFOCOM*, 2009.

[59] N. Santhapuri, Z. Zhong, and S. Nelakuditi B. Ni, "Routing with opportunistically coded exchanges in wireless mesh networks," in *Proc. of IEEE SECON*, 2006.

[60] BitTorrent. [Online]. http://www.bittorrent.com

[61] Kazza. [Online]. http://www.kazaa.com

[62] D. Smetters, J. Thornton, M. Plass, N. Briggs, R. Braynard V. Jacobson, "Networking Named Content," in *Proc. of the 5th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2009)*, Rome, Italy, Dec. 2009.

[63] R. Yates, D. Raychaudhuri, and J. Kurose S. Paul, *The cache-and-forward network architecture for efficient mobile content delivery services in the future Internet*.: ITU-T Innovations in NGN: Future Network and Services, May 2008.

[64] H. Liu, Y. Zhang, S. Paul, and D. Raychaudhuri L. Dong, "On the Cache-and-Forward Network Architecture," in *Proc. of IEEE ICC09*, June 2009.

[65] IETF RFC 3626, "Optimized Link State Routing Protocol (OLSR),".

[66] IEEE 802.11s, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 10: Mesh Networking,".

[67] D. Aguayo, J. C. Bicket, and R. Morris D. S. J. D. Couto, "A high throughput path metric for multi-hop wireless routing," in *Proc. of ACM MOBICOM*, 2003.

[68] IEEE 802.11-2007, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," , 2007.

[69] J. Prado Pavon and S. Choi, "Link adaptation strategy for IEEE 802.11 WLAN via received signal strength measurement," in *Proc. of IEEE ICC'03*, May 2003.

[70] M. Transier, C. L. M. Mauve, and W. Effelsberg B. Scheuermann, "Backpressure multicast congestion control in mobile ad-hoc networks," in *Proc. of ACM CoNEX*, 2007.

[71] The Network Simulator ns-2. [Online]. http://www.isi.edu/nsnam/ns/

[72] Ian Glover and Peter Grant, *Digital Communications*, third, Ed., 2009.