International Symposium on Information Theory and its Applications, ISITA2004
Parma, Italy, October 10–13, 2004

# Practical Schemes for Interactive Data Exchange

Giuseppe CAIRE[†] Shlomo SHAMAI[‡] Sergio VERDU[§]

[†] Institut Eurecom France [‡] Technion Israel [§] Princeton University USA

**Abstract**

Building upon Slepian-Wolf coding, sparse-graph codes, belief propagation, and closed-loop iterative doping, we propose new schemes for interactive data exchange between two agents who want to communicate losslessly their respective information via several rounds of communication.

## 1. INTRODUCTION

We consider a communication setting where remote users A and B have local data with identical length and alphabet (A has $\mathbf{x} \in \mathcal{Q}^n$, and B has $\mathbf{y} \in \mathcal{Q}^n$). After running a suitable interactive data exchange protocol including multiple rounds of bidirectional information exchange, the goal is for users A and B to obtain $\mathbf{y}$ and $\mathbf{x}$, respectively. To minimize the number of exchanged bits, the encoding/decoding protocols we propose in this paper exploit statistical dependencies between the sources that generate $\mathbf{x}$ and $\mathbf{y}$ even if those dependencies are unknown.

One motivation for this problem can be found in *content distribution networks*. Consider a network consisting of a main *gateway* and several remote *file servers* whose content is periodically updated via a high-throughput forward (broadcast) noisy link (e.g., a satellite link). For example, the content might consist of large MPEG-encoded video files that local users access after some delay.

Since the forward link is noisy, an option consists of imposing a stringent constraint on the performance of channel coding, such that post-decoding errors occur with negligible block error probability. Nevertheless, due to the size of the files involved and since the errors might have catastrophic effects on source-encoded data, the requirements of channel coding might be too constraining and eventually require too large complexity or limit the effective rate of the forward link. On the other hand, practical channel codes with very large blocklength, achieving bit-error probability of the order of $10^{-6}$–$10^{-7}$ at rates very close to the channel capacity limit are practically implementable with complexity that grows linearly with the blocklength. Suppose that we make use of such "not fully reliable" channel coding in the forward link, and that the file servers are connected to the gateway by a low-rate error-free link. (For example, they are connected via a modem to the internet.) Then, the error-free link can be exploited to correct the residual errors of the forward link by some data exchange protocol that each file server can run in conjunction with the gateway.

Another motivation for the data exchange protocol is provided by the need for maintaining consistency of remotely located files with minimal amount of data exchanged between A and B. For example, this is the goal of the popular algorithm known as RSYNC [13]. RSYNC is designed to handle "editing" operations including insertion, deletion and replication of segments of the files, as well as errors (i.e., symbol changes). In a first pass of the RSYNC algorithm, B sends to A parity-check sums generated from $\mathbf{y}$ and A determines segments of the file $\mathbf{x}$ that match segments of the file $\mathbf{y}$. Consequently, A identifies also segments of $\mathbf{x}$ that do not match any segment in $\mathbf{y}$. In a second pass, A sends to B the pointers corresponding to the matching segments and the symbols of the unmatched segments. Notice that, at the end of the first pass, A and B have agreed on a subset of segments intervals of the files $\mathbf{x}$ and $\mathbf{y}$. Sending these segments in their entirety without taking into account their correlation with the data already available at B is not efficient. Hence, our data exchange algorithms can be used to make the transmission of nonidentical but synchronized segments more efficient. We limit ourselves to handle differences between the files that are produced by the action of a standard noisy channel (not including deletions and insertions). This is in contrast to algorithms such as [13, 11, 8, 10] which are designed to handle editing operations such as deletions and insertions that destroy the synchronism between the files.

The celebrated *Slepian-Wolf theorem* [12] provides the basis for efficient schemes to perform data exchange of dependent sources of information. A conceptually straightforward (nonuniversal) principle for data exchange at the minimum possible rate is given as follows: A encodes $\mathbf{x}$ at rate $H(X|Y)$[1] and sends the compressed data to B which decodes $\mathbf{x}$ by exploiting the received compressed data from A

Partially supported by the European Network of Excellence Newcom[†‡], and by the US National Science Foundation Grant CCR-0312879[§].

[1] $H(X|Y)$ stands for the conditional entropy rate of the process $\{X_1, X_2, \ldots\}$ given $\{Y_1, Y_2, \ldots\}$

and the knowledge of $\mathbf{y}$. Knowing $\mathbf{x}$ and the joint distribution of the sources, B encodes $\mathbf{y}$ at rate $H(Y|X)$ by using arithmetic coding and sends the compressed data back to A, which can recover $\mathbf{y}$ from the knowledge of $\mathbf{x}$ and the joint distribution of the sources. The complexity bottleneck of this protocol lies in the Slepian-Wolf decoder. Slepian-Wolf compression can be implemented by linear encoders: This was shown for memoryless sources that can be written as the sum (in a finite field) of the side information and an independent process in [16][2]; general memoryless sources were encompassed in [4]; stationary ergodic sources were allowed in [14], and the optimality of linear compressors for general (nonstationary/ergodic) sources follows as a simple corollary of [2, Thm. 2.1].[3] Slepian-Wolf linear encoding reduces to computing the syndrome $\mathbf{z} = \mathbf{H}\mathbf{x}$ of the source word $\mathbf{x}$ with respect to a given parity-check matrix $\mathbf{H}$. Slepian-Wolf maximum-a-posteriori decoding corresponds to computing

$$\widehat{\mathbf{x}} = \arg \max_{\mathbf{x}\in\mathcal{Q}^n:\mathbf{H}\mathbf{x}=\mathbf{z}} P_{X|Y}(\mathbf{x}|\mathbf{y}) \tag{1}$$

Recent works such as [9, 6, 7] have considered pragmatic suboptimum decoders for linear Slepian-Wolf codes.

Clearly, the above protocol requires that the block-error probability of Slepian-Wolf decoding be arbitrarily small. Getting back to the example of the content distribution network, in this way we have just moved the requirement for small decoding error probability from the forward link to the data exchange protocol, which is not an easier task in general.

In this paper we propose practical low-complexity embodiments of Slepian-Wolf compression and decompression which capitalize on the interactive and bidirectional nature of the communication between A and B. Our first protocol involves multiple rounds, has variable length, achieves arbitrarily low probability that the files reproduced at A and B are not identical to the actual files, and achieves rates slightly above $H(X-Y)$ in both directions (from A to B and viceversa). In the case where the files have no redundancy this algorithm is almost optimal. We also offer two other alternatives which can achieve better efficiency if the sources have high redundancy.

## 2. DATA COMPRESSION WITH LINEAR CODES

In [2] we have presented a variable-length scheme for universal data compression of sources with memory. Compression is achieved by three main components

- Block Sorting
- Syndrome calculation
- Closed Loop Iterative Doping

For the purposes of the present paper, we will focus on syndrome calculation and closed loop iterative doping. To compute the syndrome, the string to be compressed is multiplied by a rectangular matrix which is the parity-check matrix of an error control code designed for an additive-noise discrete channel whose noise has the same statistics as the data to be compressed.

Since the complexity of maximum-a-posteriori decoding is in general not feasible, it is natural, as proposed in [2] to use low-density parity-check matrices in conjunction with belief propagation decoding. However, the state of the art in code design does not yield low block error rate when the compression rate is only slightly above capacity.

To overcome this shortcoming, [3, 2] came up with the *closed-loop iterative doping* (CLID) algorithm which works in conjunction with the belief propagation decoder. After the syndrome has been transmitted, the encoder runs an exact copy of the belief propagation algorithm that is run at the decoder and therefore, at each iteration, is able to monitor the reliability of all the source symbols at each iteration in the belief propagation algorithm. By supplying to the decompressor the lowest-reliability symbol (uncompressed) at certain preagreed iterations, the CLID is able to supply the decoder with information which is maximally useful for the belief propagation decoder to converge. This is done efficiently since, conditioned on the data received so far, the supplied symbol has essentially maximum entropy.

## 3. BIDIRECTIONAL S-W SCHEMES

### 3.1. Algorithm A

We present the nonuniversal version of our data exchange algorithm which assumes knowledge of the first-order entropy of the difference between both sources of information, namely

$$H = H(X_i - Y_i).$$

where the entropy is measured in $\log_{|\mathcal{Q}|}$ units. Note that in a universal setting in which the statistics of the difference process are a priori unknown, both encoders can learn the first-order statistics of $X_i - Y_i$ and thus agree on a value of $H$ at the expense of a negligible rate increase in bidirectional communication.

Let $\epsilon > 0$ be a design parameter. Suppose that a parity-check matrix $\mathbf{H}$ whose ratio of rows to columns is equal to $H + \epsilon$, is designed for an additive-noise discrete channel whose noise is memoryless with distribution identical to that of $X_i - Y_i$.

[2]This fact played a key role in the inception of the Slepian-Wolf result (J. K. Wolf, private communication)

[3]This is in contrast to channel coding, where linear codes are optimal in the domain of additive-noise discrete memoryless channels.

The first step in the information exchange is for A to communicate the "syndrome" vector $\mathbf{Hx}$ to B and for B to communicate $\mathbf{Hy}$ to A.

Each location then subtracts the received vector from the syndrome vector it had computed to obtain identical copies of the vector $\mathbf{H}(\mathbf{x}-\mathbf{y})$.

In the second step, $\mathbf{H}(\mathbf{x}-\mathbf{y})$ is used at both locations to run identical copies of belief propagation on the Tanner graph of $\mathbf{H}$ where the check nodes have values given by $\mathbf{H}(\mathbf{x}-\mathbf{y})$ and the variable nodes start with the a priori values given by the a priori distribution of $X_i - Y_i$.

At each iteration $i$ (or at a preselected subsequence of iterations) both identical copies of the belief propagation algorithm determine the location, $\ell_i$, of the lowest-reliability symbol and they exchange the value of $\mathbf{x}_{\ell_i}$ and $\mathbf{y}_{\ell_i}$. Both locations then compute $\mathbf{x}_{\ell_i} - \mathbf{y}_{\ell_i}$ and henceforth fix the reliability of the $\ell_i$ symbol to $\infty$. Thus, as in the scheme outlined in Section 2, even though raw symbols are exchanged, they have almost no redundancy conditioned on the information available up to that point.

The iterations terminate when the most likely vector of differences $\hat{\mathbf{e}}$ computed by the belief propagation algorithm satisfies $\mathbf{H}(\mathbf{x}-\mathbf{y}) = \mathbf{H}\hat{\mathbf{e}}$.

It can be seen that

$$\max\{H(X|Y), H(Y|X)\} \leq H(X_i - Y_i) \qquad (2)$$

with equality if both sources have no redundancy, and the differences $X_i - Y_i$ are iid. Thus, in that case, as the blocklength increases the scheme achieves the Slepian-Wolf limit. Otherwise it suffers a penalty in efficiency. Note that for the relevant case of the satellite content distribution network the scheme is indeed almost optimal, as in that case the MPEG files have almost no redundancy and after the forward link decoding, errors are almost independent if long-blocklength codes are used, as for example in the DVB-S2 next-generation satellite video broadcasting, based on an LDPC code of length 64800.

The intimate relation between the Slepian-Wolf concept governing efficient interactive data exchange and cryptographic key generation in the presence on an eavesdropper that monitors the public channel, is by now very well recognized [1]. In fact the specific interactive data exchange model we examine here where A observes $\mathbf{x}$ and B observes a correlated version $\mathbf{y}$, and both exchange data over a public channel, is a classical model to generate a secret key. In this case the eavesdropper monitoring the communication over the public channel must remain completely ignorant of the content of the secret key. This describes exactly the Model S (source-type model) in the terminology of [1], where it is established that the secret key capacity is equal to the mutual information between the sources generating $\mathbf{x}$ and $\mathbf{y}$. Algorithm A may be used to approach this ultimate limit, as is demonstrated, for example, by the case where both sources generate correlated fair coin flips $\mathbf{y} = \mathbf{x} + \mathbf{e}$, and $\mathbf{e}$ is Bernoulli with parameter $p$. An eavesdropper monitoring the communications over the public channel has access to $\mathbf{Hx}$, $\mathbf{Hy}$ and the $d$ doped bits generated by CLID in each direction. Assuming that $\mathbf{x}$ is used to generate the secret key, the secret key rate generated by the scheme (private rate of information not available to the eavesdropper) is equal to $(n - (H+\epsilon)n - d)/n$ which is approximately equal to the ultimate limit $1 - h(p)$. Note that the eavesdropper gets $(H+\delta)n$ bits about $\mathbf{x}$, but since the entropy of $\mathbf{x}$ is $n$, it is able to generate the secret key with very small probability.

Algorithm A can be modified to broadcast scenarios where a server containing the "true" information updates a plurality of sites each containing corrupted information. If the server sends the same information to all sites, it is convenient to replace the LDPC code by a fountain code, so that sites that have less corrupted versions can terminate their protocol faster. Then, private rounds of CLID with each site can be conducted as explained above.

### 3.2. Algorithm B

The symmetric nature of Algorithm A may be dictated in some applications in which communication in both directions takes place simultaneously; however, in those applications in which the rounds of communications happen sequentially, it is possible to increase efficiency by means of the following three stage process. (Throughout this description, the role of A and B, and in particular, which agent initiates the communication, is at the disposal of the designer.) In the first stage, the data transfer occurs from B to A using a linear code with rate high enough to enable A to recover (even in the absence of CLID), $\hat{\mathbf{y}}$, a noisy version of $\mathbf{y}$. In many cases, a good model for the difference between $\hat{\mathbf{y}}$ and $\mathbf{y}$ is

$$\hat{\mathbf{y}} = \mathbf{y} + \hat{\mathbf{e}} \qquad (3)$$

where $\hat{\mathbf{e}}$ is independent of $\mathbf{y}$. In the second stage, Algorithm A is applied with $\hat{\mathbf{y}}$ taking the role of $\mathbf{x}$. Note that if indeed the bit error rate in the first decoding step (at A) was low, $H(\hat{\mathbf{e}})$ is small, and the communication requirements from A to B in the second stage can be made to be quite low in order for A to reach a perfect copy of $\mathbf{y}$. Further communication savings can be realized by noticing that since there has already been communication from B to A in the first stage, the parity check equations chosen in the embodiment of Algorithm A can be simply a subset of those chosen at the first stage, thereby obviating any further traffic from B to A except for doped bits generated by CLID. In the third stage, all that remains is for A to communicate $\mathbf{x}$ to B but since both encoder and decoder know $\mathbf{y}$ exactly as well as the joint distribution this can be done efficiently using slightly more than rate $H(X|Y)$ (e.g., by using standard arithmetic coding). Note that a further source of asymmetricity in some

content distribution applications is the fact that only A must be informed of B but not viceversa, in which case the third stage can be obviated.

An interesting design issue arises in this protocol, as frequently $P[\hat{\mathbf{e}} = \mathbf{0}]$ is close to 1. In this case, in situations where several independent blocks of data are encoded in separate blocks from B to A, it is advantageous to encode the corresponding information from A to B not on a codeword-per-codeword basis but taking into account the concatenation of several consecutive blocks (with interleaved bits across the blocks) In this way, we can approach the ergodic behavior of the channel (3).

### 3.3. Algorithm C

When the files are very redundant due to memory, the sides in (2) can be quite different. In that case, it is of interest to consider a more efficient algorithm. We can follow the same first step as before encoding with LDPCs of rate slightly larger than $H(X_i - Y_i|Y_1^n)$ from A to B and $H(X_i - Y_i|X_1^n)$ from B to A. Then decode with belief propagation run to obtain the difference $(\mathbf{x} - \mathbf{y})$. The decoders require knowledge of the conditional distributions $P_{X_i|Y_1^n}$ and $P_{Y_i|X_1^n}$. Unless these are identical, it is no longer feasible to run identical copies of belief propagation at both locations, and thus CLID is infeasible. (However, low reliability symbols can still be interchanged at the expense of having to identify their locations.) Furthermore, since in most cases the conditional distributions are unavailable or infeasible to compute, an attractive implementation uses the DUDE+ algorithm [15] in order to estimate $P_{X_i|\mathbf{y}}$ at B and $P_{Y_i|\mathbf{x}}$ at A, by running DUDE+ with the data present at each site as the noisy output of a discrete memoryless channel whose transition probability matrix is given by the conditional marginals $P_{X_i|Y_i}$ at A and $P_{Y_i|X_i}$ at B.

### 3.4. Algorithm D

With the algorithm in the preceding paragraph there is still a penalty relative to the Slepian-Wolf-Cover rate equal to the limit of

$$\begin{aligned} & \frac{1}{n}\sum_{i=1}^{n} H(X_i - Y_i|Y_1^n) - \frac{1}{n}H(X_1^n|Y_1^n) \\ = \quad & \frac{1}{n}\sum_{i=1}^{n} I(X_i; X_1^{i-1}|Y_1^n) \end{aligned} \tag{4}$$

from A to B, and analogously, interchanging the roles of $X$ and $Y$, from B to A. Based on the equivalence of Slepian-Wolf decoding and joint source-channel decoding we can overcome the penalty in (4) if $X_i$ is Markov and the channel is memoryless (or more generally when $(X_i, Y_i)$ is Markov), both with known statistics.

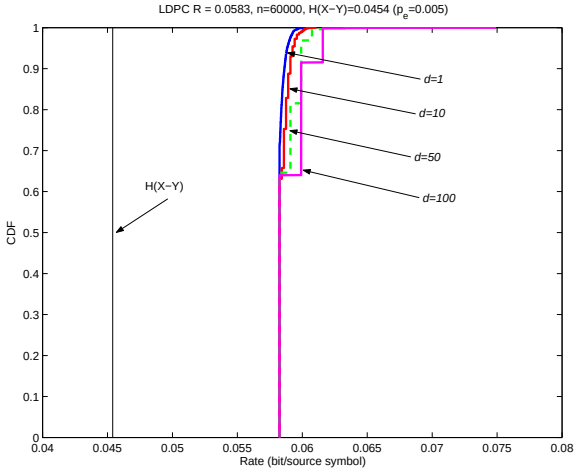


Figure 1: Empirical cdf of Algorithm A rate for different values of the CLID blocklength $d$.

In Algorithm D an LDPC with matrix $\mathbf{H}$ of rate slightly larger than $H(X|Y)$ is used to communicate the syndrome vector $\mathbf{Hx}$ from A to B. Then, a graphical model in which the variable nodes correspond to $X_1^n$ is set up incorporating both the Tanner graph of $\mathbf{H}$ and the Markov structure of $X_1^n$. To obtain $\mathbf{x}$ with high probability a belief propagation algorithm is run on the combined graph incorporating efficient forward-backward dynamic programming recursions and message passing on the Tanner graph. A parallel, but noninteracting, scheme is run in the reverse direction.

### 3.5. Algorithm E

Another scheme to deal with sources with memory can be obtained using Huffman codes. In this scheme, it is beneficial to group letters into equal-length blocks which are regarded as 'supersymbols.' Henceforth, those supersymbols take the role of the symbols of the vectors $\mathbf{x}$ and $\mathbf{y}$. Suppose that $P_{X_i|X_1^{i-1},Y_1^{i-1}}$ and $P_{Y_i|X_1^i,Y_1^{i-1}}$ are available at both A and B. Assume also that at stage $i$ A has $y_1....y_{(i-1)}$ and B has $x_1...x_{(i-1)}$. A encodes $x_i$ using a Huffman code with distribution $P_{X_i|X_1^{i-1},Y_1^{i-1}}$ and sends it to B. Upon receipt of this information from A, B uses the Huffman decoder for the code that A just used to decode $x_i$. With that value and the existing $x_1...x_{(i-1)}$, B uses $P_{Y_i|X_1^i,Y_1^{i-1}}$ to construct a Huffman code to encode $y_i$. It then sends this encoded value to A, which decodes $y_i$. Now A has $y_1....y_i$, B has $x_1....x_i$ and the process repeats. The required communication is

$$\mathsf{A} \to \mathsf{B}: \qquad \sum_{i=1}^{n} H(X_i|X_1^{i-1}, Y_1^{i-1}) + a_i \tag{5}$$

$$\mathsf{B} \to \mathsf{A}: \qquad \sum_{i=1}^{n} H(Y_i|X_1^{i}, Y_1^{i-1}) + b_i \tag{6}$$

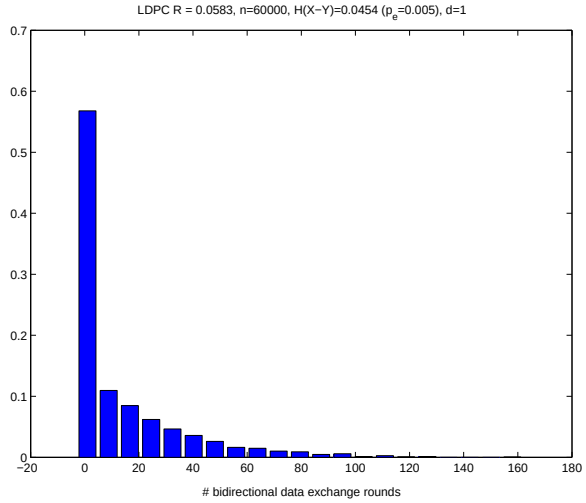


Figure 2: Histogram of the number of bidirectional data exchanges for Algorithm A, $d = 1$, in the setting of Fig.1.

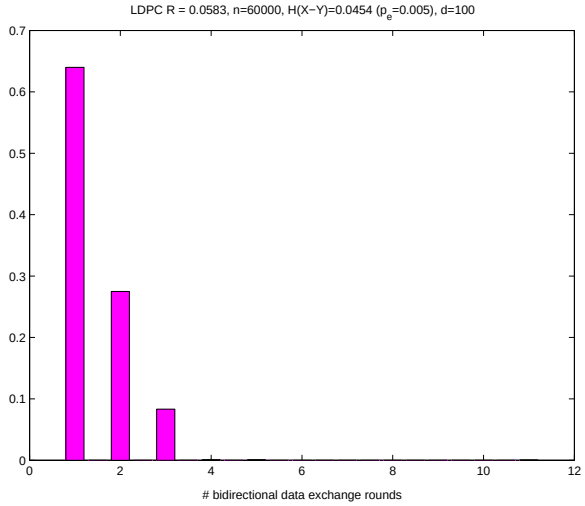


Figure 3: Histogram of the number of bidirectional data exchanges for Algorithm A, $d = 100$, in the setting of Fig.1.

where $0 \leq a_i \leq 1$ bits, $0 \leq b_i \leq 1$ bits. Thus from B to A there is a loss of efficiency of $I(X_{i+1}^n; Y_i | Y_1^{i-1}, X_1^i) + b_i$ bits at each stage, while from A to B the loss of efficiency is: $I(Y_i^n; X_i | X_1^{i-1}, Y_1^{i-1}) + a_i$ bits at each stage. Note that an interesting problem is to determine the length of the super-symbol that will be be most efficient. In order to make the scheme universal, we can run a context modeler like the one required in arithmetic coding to estimate $P_{X_i, Y_i | X_1^{i-1}, Y_1^{i-1}}$ from the decoded data.

## 4. EXPERIMENTS

We consider first Algorithm A. Let $\mathbf{x}$ consist of fair coin flips and $\mathbf{y} = \mathbf{x} + \mathbf{e}$, with $\mathbf{e}$ Bernoulli-$p_e = 0.005$, yielding $H(X_i - Y_i) = 0.0454$ bit/symbol. An irregular LDPC ensemble with blocklength $n = 60000$ and degree sequences $\lambda(x) = 0.3479x^3 + 0.3178x^{10} + 0.0386x^{31} + 0.2957x^{32}$, $\rho(x) = 0.5x^{108} + 0.5x^{109}$ is generated without any effort to select individual good instances. The compression coding rate is 0.0583, corresponding to a 28% relative gap from the the Slepian-Wolf limit $0.0454$. Naturally, a careful code optimization can narrow this gap. However, since this is equivalent to the problem of designing codes for the BSC, we choose to focus our simulation effort in illustrating the behavior of the CLID. At each bidirectional data exchange round, a block of $d$ CLID bits are sent. We considered the cases $d = 1, 10, 50$ and 100. Fig.1 shows the protocol aggregate rate cdf over 4000 trials, for the different values of $d$. The BP decoder was run for 100 initial iterations, and then a CLID data block is exchanged every 20 iterations, until convergence is reached. We assumed that the terminals agree on convergence if all parity-check equations are satisfied. The event that the parity-check equations are satisfied but the reconstructed sequence is wrong never occurred in any of our experiments.[4] Figs. 2 – 3 show the empirical distribution of the number of bidirectional data exchange rounds for the different values of $d$ used. We notice that there is a very interesting tradeoff between the communication rate in terms of bit/source symbol and the number of protocol rounds. Larger $d$ yields a more wide rate distribution (and a larger average rate), but fewer bidirectional rounds. In practical networking applications, the option of large $d$ might be preferred. For example, we notice here that with $d = 100$ the protocol normally requires between 1 and 3 rounds. Moreover, in certain applications exchanging one bit or a packet of $d$ bits takes essentially the same effort because of the various protocol layers overhead. Hence, reasonably large values of $d$ are to be preferred. In our second example, we experiment with Algorithm B. In this case, $\mathbf{y}$ is Bernoulli-$p_y = 0.05$. Terminal A has a noisy version $\mathbf{x} = \mathbf{y} + \mathbf{e}$ of the desired source sequence, where $\mathbf{e}$ is Bernoulli-$p_e = 0.1$ independent of $\mathbf{y}$. We have $p_x = p_e p_y + (1 - p_e)(1 - p_y)$ and

$$H(Y|X) = (1 - p_x) h\left(\frac{p_e p_y}{1 - p_x}\right) + p_x h\left(\frac{(1 - p_e) p_y}{p_x}\right)$$

yields $H(Y|X) = 0.1712$. Notice that with straightforward application of Algorithm A only the much higher rate $H(X - Y) = h(p_e) = 0.469$ can be achieved. (Note that this is the optimal rate to convey $\mathbf{x}$ to B.) We have designed an irregular LDPC ensemble with compression rate $R = 0.2$, blocklength $n = 60000$ and degree sequences $\lambda(x) = 0.0967x^2 + 0.1778x^3 +$

[4] If needed, a CRC might be inserted as in standard Internet protocols in order to improve the probability of the error event.

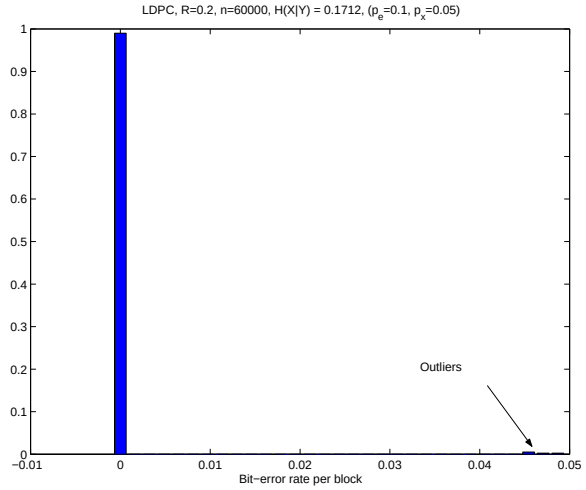


Figure 4: Histogram of the bit-error rate per block for step 1 of Algorithm B.

$0.1709x^7+0.0132x^8+0.0674x^9+0.0373x^{21}+0.0683x^{22}+0.1060x^{28}+0.0674x^{29}+0.1102x^{95}+0.0848x^{96}$, $\rho(x) = 0.5x^{33}+0.5x^{34}$. Using this code in step 1 of Algorithm B (i.e., B sends $\mathbf{Hy}$ to A and A decodes $\widehat{\mathbf{y}}$ using the full knowledge of $\mathbf{x}$), yields an average residual bit-error probability $\widehat{p} = 5 \cdot 10^{-4}$.

Fig. 4 shows the histogram of the block-by-block bit-error probability (number of erroneous bits divided by the blocklength) after step 1 decoding. We notice that the bit-error probability is very small (around 4 $10^{-6}$) with high probability, but for a small number (around 1%) of *outliers* yielding bit-error probability lightly less than $0.05$. In any case, even assuming block-by-block decoding, the scheme of the previous example can be used for step 2 of Algorithm B (working with $p_e = 0.05$). The total rate achieved from B to A (aggregate of steps 1 and 2) is then $0.2583$ bits/symbol. If interleaving among blocks is used, the rate due to step 2 can be reduced further. The rate in the reverse direction is only about 0.0583 bit/symbol.

## References


[1] R. Ahlswede and I. Csiszár. Common randomness in information theory and cryptography-part I: Secret sharing. *IEEE Trans. Inform. Theory*, 39, No. 4:1121–1132, July 1993.

[2] G. Caire, S. Shamai, and S. Verdú. Noiseless data compression with low density parity check codes. In P. Gupta and G. Kramer, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 2004.

[3] G. Caire, S. Shamai, and S. Verdú. A new data compression algorithm for sources with memory based on error correcting codes. *2003 IEEE Workshop on Information Theory*, pages 291–295, Mar. 30- Apr. 4, 2003.

[4] I. Csiszár. Linear codes for sources and source networks: Error exponents, universal coding. *IEEE Trans. Inform. Theory*, IT-28:585–592, July 1982.

[5] P. Elias. Coding for noisy channels. *IRE Convention record*, 4:37–46, 1955.

[6] J. Garcia-Frias and Y. Zhao. Compression of correlated binary sources using Turbo codes. *IEEE Communication Letters*, 5:417–419, Oct. 2001.

[7] A. D. Liveris, Z. Xiong, and C. N. Georghiades. Compression of binary sources with side information at the decoder using LDPC codes. *IEEE Communication Letters*, 6, no. 10:440–442, Oct. 2002.

[8] Y. Minsky, A. Trachtenberg, and R. Zippel. Set reconciliation with nearly optimal communication complexity. *2001 IEEE Int. Symp. Information Theory*, page 232, June 2001.

[9] T. Murayama. Statistical mechanics of the data compression theorem. *J. Phys. A: Math. Gen.*, 35:L95L100, 2002.

[10] A. Orlitsky and K. Viswanathan. One-way communication and error-correcting codes. *2002 IEEE Int. Symp. Information Theory*, page 394, July 2002.

[11] A. Orlitsky and K. Viswanathan. Practical protocols for interactive communication. *2001 IEEE Int. Symp. Information Theory*, page 115, June 2001.

[12] D. Slepian and J. K. Wolf. Noiseless coding of correlated information sources. *IEEE Trans. Information Theory*, IT-19:471–480.

[13] A. Tridgell. *Efficient Algorithms for sorting and synchronization*. PhD thesis, The Australian National University, Canberra, ACT, Australia, 2000. See also rsync.samba.org.

[14] T. Uyematsu. An algebraic construction of codes for Slepian-Wolf source networks. *IEEE Trans. on Information Theory*, 47, no. 7:3082–3088, Nov. 2001.

[15] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdú, and M. Weinberger. Universal discrete denoising: Known channel. In *IEEE Symposium on Information Theory*, 2003.

[16] A. D. Wyner. Recent results in the Shannon theory. *IEEE Trans. Information Theory*, IT-20, Jan. 1974.