Reflections on the Prize Paper:

"Near optimum error-correcting coding and decoding: turbo codes"

Claude Berrou and Alain Glavieux
Ecole Nationale Supérieure des Télécommunications de Bretagne, Brest, France

Introduction
What is now commonly called the information age began with a double "big bang", and this year we are celebrating its fiftieth anniversary. For it was in 1947-48, at a few months' interval and in the same institution (Bell Labs, New Jersey, USA), that William Shockley and his team invented the transistor and Claude Shannon published his theory of communications. This prodigious coincidence saw the twin birth of the semiconductor component, which can represent binary information by its physical state, and the "bit", the mathematical unit measuring this information. Today, we can better appreciate the real importance of these two discoveries which have deeply marked this half-century technically, socially and even politically. So it is a great honor for us to have been invited in this particular year to write these reflections.

The article is divided into two distinct parts. The first part describes the progression, experimentation, observation and reflection which led to a new family of error-correcting codes: turbo codes. What will be noticed in this chronologically narrated experience is the naiveté required to invent this new coding scheme and its associated decoder. In the second part, specific (self-concatenated) turbo codes are presented and the usual turbo decoding rules are summarized.

PART I

The context
The "Ecole Nationale Supérieure des Télécommunications de Bretagne" was created by France Télécom in 1978. After some years spent setting up the teaching program, research activities began in electronics, signal processing and computer science. The mainly young researchers were free to define their own fields of investigation, in collaboration with other research centres (CNET, CCETT, ...) in Brittany, a region where menhirs, church steeples and telecommunications antennae share the skyline. With our colleague Patrick Adde, we decided to focus our attention on the Soft-Output Viterbi Algorithm (SOVA).

Soft-output Viterbi decoding
Two articles [1,2] marked the beginning of our work on error-correcting codes, not forgetting of course inescapable references like [3,4]. Our aim was to translate the SOVA algorithm described in [1] into MOS transistors, in the simplest possible way. We met Gérard Battail several times and it appeared that there was definitely much to be done in and perhaps something to contribute to this fascinating area of coding. We eventually arrived at a suitable hardware solution [5]. This took a long time

Continued on page 24

1CNET: Centre National d'Études de Télécommunications, CCETT: Centre Commun d'Études de Télédiffusion et Télécommunications.
Reflections on the Prize Paper
continued from page 1

3/4 and 2/3 (Fig. 1), we thought that it would be interesting to see what would happen if we reinjected the result of the outer decoding into the inner decoder. As the different levels of the composite decoder do not represent the same pieces of information (the codes are not systematic), it was necessary to build an estimate of symbols $x_2$ and $y_1$ at the output of SOVA$_2$. At first, it was a great surprise to observe that the bit error rate (BER) of these reconstructed symbols after decoding was lower than that of decoded information $d$. We were unable to find any explanation for this strange behavior in the literature. It was then a straightforward task to (re)invent recursive systematic convolutional (RSC) codes to take advantage of this heretofore unreported property. We gave Punya Thitimajshima, one of our Ph.D. students, the work of analyzing the distance properties of these codes [8].

**Extrinsic information**

The soft-output Viterbi decoder of a systematic code provides a good estimate of the log likelihood ratio (LLR) relative to any input symbol, which can obviously be seen as the sum of two contributions. The first, intrinsic information (possibly including a priori probability, see part II), is already available before decoding; the second, extrinsic information, is provided by exploiting the dependencies (due to convolution, parity, ...) which exist between this particular symbol and the other symbols processed by the decoder. In any de-

**Concatenation**

Using our version of the SOVA, we were able to cascade SNR amplifiers and do the experiments described in [6], which was our initial plan. That is, we decoded a classical (i.e. serial) concatenation of two ordinary (i.e. nonsystematic, nonrecursive) convolutional codes or even more than two. Concatenation is a simple way to obtain large asymptotic gains [7], but the performance at low SNRs is degraded due to having to share the redundancy energy between the different component codes.

**Feedback and recursive systematic convolutional codes**

Having simulated the decoding of several simple concatenation schemes, typically two serial codes with rates...
coding construction, extrinsic information must not be used by the processor which produced it. In a way, this principle had already been stated by Gallager in [9], and in parallel with our work, it was also applied by Lodge et al. [10]. We came up with the scheme depicted in Fig. 2. The decoder is built in such a way that no information produced by a component decoder can feed its input, either directly or indirectly.

**Iterative decoding**

The digital processing of information has at least one major drawback: it does not lend itself easily to feedback techniques. An iterative procedure has to be used because of the delays (trellis, interleaving, ...). This procedure increases the decoder latency, but ever-continuing progress in microelectronics makes possible today what was unimaginable not so long ago.

Two hardware solutions can be contemplated, depending on the information rate. For low data rates, a single decoding processor working at a high frequency can make all the required iterations with a tolerable added delay. For high rates, a cascade of identical decoding modules can be implemented monolithically to enable high-speed pipe-line processing (in this case, typically that of broadcasting, the latency problems are generally less crucial).

**Parallel concatenation**

The idea of so-called parallel concatenation did not germinate by analogy with product codes. More trivially, we were seeking to simplify the clock distribution in our envisaged high-data-rate integrated circuit, because classical (serial) concatenation requires different clocks to be generated. Parallel concatenation (Fig. 3) simplifies the architecture of the composite decoder as the two encoders and the two component decoders work with the same clock.

Having said that, what is the general principle of error-correcting coding? It is the distribution of the energy available between different redundant symbols, in such a way that the receiver can take advantage of a diversity effect; the greater the number of redundant symbols, the greater the diversity effect. With the same coding rate, and hence the same

---

3Perhaps analog electronics will remove this handicap one day [11,12].
transmitting bandwidth, parallel concatenation yields more redundant symbols from the outer code than does serial concatenation. Therefore this coding construction was not inappropriate [13].

The results of our first simulations were both encouraging and frustrating: encouraging because we had unusually good performance at low SNRs; frustrating, because the BER had difficulty in falling below $10^{-5}$.

**BER floor**

Unlike serial concatenation, parallel concatenation requires the decoder to take particular care over the importance to be given to the different data available. Thus, in the decoder in Fig. 3.b, the reliability of $LLR_i(d)$ increases with the number of iterations, whereas that of $Y_2$ always remains the same. Therefore we must take this evolution into careful consideration to avoid the BER stagnating at a value imposed by the noise affecting $Y_2$ [14]. The same is true for the relative reliabilities of $X_1$, $Y_1$, and $Z$, but this is apparently less important. That is why we now prefer to use the parallel decoder as depicted in Fig. 3.c, which eliminates the weighting problem for $Y_2$. Finally, by replacing the SOVA by the BCJR (APP) algorithm [15], and by manipulating probabilities instead of LLRs [16], weighting problems are removed.

**Non-regular interleaving**

Another possible cause of BER flattening is insufficient minimum code distance. We examined closely the reasons for the error patterns (regular, much more often than not) that the turbo decoding of two parallel concatenated codes could not eliminate. It seemed clear that some disorder could be advantageously introduced in the interleaving permutation. Pseudo-random transposition is a concept of prime importance in cryptography where the greatest distance is sought between the original message and the encrypted message. In a way, the approach is the same with error-correcting coding; it is a quite fascinating concept, associating as it does algebra, geometry and statistics.

With serial concatenation, regular interleaving is generally sufficient to obtain large minimum distances and asymptotic gains, but the convergence threshold of iterative decoding moves away from the Shannon limit. Perhaps a composite code combining the advantageous properties of each concatenation scheme will be found before long [17].

**Conclusion to Part I**

Turbo codes are the result of an empirical, painstaking construction of a global coding/decoding scheme, using existing bricks that had never been put together in this way before. This type of construction was in the air, and other schemes are sure to come with other conceptual approaches. "It is often said that one should experiment without having any preconceived ideas. That is not possible: each person carries within himself his conception of the world that he cannot easily put aside" (H. Poincaré).

**PART II**

In the article originally published at ICC’93, the encoder was made from the parallel concatenation of two RSC codes and an interleaver, as shown in Fig. 3. Although this encoder is well-adapted for continuous data transmission, it poses a few problems when used for block transmission because of the unknown final state after encoding the block. This is what led us to modify its structure, in such a way that the state of the encoder at the end of each block should always be known, without having to add any termination information. This new encoder does of course have properties fairly similar to those of parallel concatenation.

Before presenting this encoder, hereafter called the self-concatenated block encoder, we shall first demonstrate that parallel concatenation has some analogies with pseudo-random coding.

**Some reflections on parallel concatenation**

Let us consider a 1/3 rate encoder built from two RSC codes, $C_1$ and $C_2$, with the same polynomial generators $G_1(D)$ and $G_2(D)$, having values of 15 and 17 in octal notation.

$$G_1(D) = 1 + D + D^3$$
$$G_2(D) = 1 + D^2 + D^3$$  \(1\)

where $D$ is the delay variable and the additions are modulo 2.

Calling $X(D)$ and $Y_1(D)$ the $D$-transforms of the sequences $\{X_i\}$ and $\{Y_1,i\}, k$ being the time index, and taking into account the recursive nature of the codes used, we have:

$$Y_1(D) = X(D) \frac{G_2(D)}{G_1(D)}$$  \(2\)

The parity check equation that code $C_1$ satisfies at each time $k$ is:

$$X_1 + X_{k-1} + X_{k-2} + X_{k-3} + Y_{1,k} + Y_{1,k-1} + Y_{1,k-2} = 0$$  \(3\)

The parity check equation associated with code $C_2$ has an expression analogous to (3) except that for a given time $n$, the indices of data symbols are not adjacent.

$$X_{i(n)} + X_{j(n)} + X_{l(n)} + X_{p(n)} + Y_{2,n} + Y_{2,n-1} + Y_{2,n-2} = 0$$  \(4\)

where the indices $i, j, l$ and $p$ depend on the instant $n$ according to the law of interleaving used.

The two parity-check equations (3) and (4) together characterize the composite 1/3 rate code. To an observer who does not know the interleaving law, it might appear that these two checks are statistically independent. In fact, however, the information symbols in (4) are the same up to permutation as those in (3), so that the composite code can be regarded as a single time-varying code with a very long constraint length depending on the permutation law.
Structure of the self-concatenated block encoder

We shall first recall some of the properties of an RSC code. The recursive part of the encoder is defined by a polynomial $G_1(D)$ of the form:

$$G_1(D) = 1 + \sum_{i=1}^{n} g_i D^i \quad g_i \in \{0,1\}$$  \hspace{1cm} (5)

Let us assume that the encoder is initially in any state. After introducing a certain number of data symbols $d_k$ equal to zero, the encoder, behaving like a pseudo-random generator, will return to its initial state. This number of "0"s is called the encoder period and denoted $L$. It can be shown that $L$ is equal to the smallest degree of any polynomial of the form $1 + D^r$, that is a multiple of $G_1(D)$. For example, since $1 + D^4$ is the polynomial $1 + D + D^2 + D^3 + D^4$ with the smallest degree $r$ that is a multiple of $G_1(D) = 1 + D + D^2 + D^3 + D^4$, the period of the recursive encoder of polynomial $G_1(D)$ is $L = 5$.

Let us now assume that the initial state of the encoder is $(00\ldots0)$. Hence, as shown in [18], any $(d_k)$ sequence made up of a symbol "1" followed by $(pl - 1)$ symbols "0" $(p = 1,2,\ldots)$ then, finally by a symbol "1", will enable the encoder to return to its initial state (this is only true if the initial state is $(00\ldots0)$). In practice, the sequence at the input of the encoder does not, of course, satisfy the previous condition. However, by judiciously associating a memory and an RSC encoder, it is possible to bring the encoder into the zero state at the end of each block.

Let us consider the block transmission of $K$ data symbols $d_k$, where $K$ is a multiple of $L$. Each symbol $d_k$ is stored twice in a memory of size $2K$, the first time at an address $1 \leq n \leq K$ and the second time at an address $n + p_jL$ where $p_j$ is an integer such that $K + 1 \leq n + p_jL \leq 2K$. In this way, the sequence feeding the encoder can be regarded as the sum of $K$ basic sequences of length $2K$ of the form given in Fig. 4. If the encoder is initially in the state $(00\ldots0)$, then it will be in this same state again at the end of each basic sequence. Since the encoder is linear and any sequence is composed of a sum of basic sequences, the final state of the encoder (the sum of the basic final states) will therefore always be $(00\ldots0)$.

This block encoder (Fig. 5) thus produces codewords $C$ made up of $K$ data bits $d_k$ and $2K$ redundancy bits $c_k$ which can if necessary be punctured in order to obtain the desired coding rate.

This self-concatenated encoder obviously has many analogies with the encoder built from the parallel concatenation of two RSC codes. For these two encoders, each bit $d_k$ produces two redundancy bits, at different instants. The optimal decoding of this block code thus poses the same problems as the code originally considered, when the length of the blocks exceeds several tens of units. However, by using iterative or turbo decoding, it is possible to approach optimal decoding performance very closely.

Soft-input/soft-output decoders

Since a turbo decoder is built from soft-input/soft-output (SISO) elementary decoders, let us briefly recall the principle of these decoders.

Let us consider the general case of a systematic code, recursive or not, which associates a codeword $C$ of length $N$ to each block of $K$ information bits $d_k$. Each codeword is transmitted on a binary-input channel with additive white Gaussian noise, which gives an observation $R_i^k$ composed of $N$ samples of the form:

$$r_i = (2c_i - 1) + n_i \quad \text{if } 1 \leq i \leq K$$
$$r_i = (2c_{i-K} - 1) + n_i \quad \text{if } K + 1 \leq i \leq N$$  \hspace{1cm} (6)

where the $c_i$ represent the redundancy symbols and the $n_i$ are the noise samples with variance $\sigma_n^2$.

A soft-output decoder associates a decision to each bit $d_k$ in the form of an $a posteriori$ probability $APP_i(d_k)$ or a log-likelihood ratio $LLR_i(d_k)$ with:

$$APP_i(d_k) = \Pr\{d_k = i \mid R_i^N\}$$  \hspace{1cm} (7)

$$LLR_i(d_k) = \log \frac{\Pr\{d_k = i \mid R_i^N\}}{\Pr\{d_k = 0 \mid R_i^N\}}$$  \hspace{1cm} (8)

By comparing either of these two quantities with a threshold fixed at $1/2$ or $0$ respectively, a hard decision on $d_k$ can be obtained. The reliability of this decision is a function of the amplitude of $APP_i(d_k)$ or of $|LLR_i(d_k)|$.

Let us first consider the $a posteriori$ probability, which can also be written as follows:

$$APP_i(d_k) = \sum_{c_k \in C_k^i} \Pr\{C = c_k \mid R_i^N\}$$  \hspace{1cm} (9)

where $c_i$ is a specific codeword and $C_k^i$ represents the
set of codewords for which \( d_k = i \) (\( i = 0, 1 \)). By using Bayes' rule, relation (9) becomes:

\[
\text{APP}_i(d_k) = \frac{1}{p(R^N_i)} \sum_{c_l \in C} p(R^N_i | C = c_l) \text{Pr} \{ C = c_l \}
\]

Each codeword containing \( K \) data bits, assumed to be mutually independent, we can write:

\[
\text{Pr} \{ C = c_l \} = \prod_{n=1}^{K} \text{Pr} \{ d_n = d_n^l \} = \prod_{n=1}^{K} \pi_{d_n^l}
\]

where \( d_n^l \) is the value of the symbol \( d_n \) of the codeword \( c_l \), and \( \pi_{d_n^l} \) is its \textit{a priori} probability. By taking into account the fact that the components of the observation are mutually independent given the transmitted codeword, the \textit{a posteriori} probability is finally equal to:

\[
\text{APP}_i(d_k) = \frac{1}{p(R^N_i)} \sum_{c_l \in C} \prod_{n=1}^{K} p(r_n | d_n^l) \pi_{d_n^l} \prod_{n=K+1}^{N} p(r_n | c_n)
\]

where \( \pi_n = \text{Pr}(d_n = i) \). As expression (12) shows, the \textit{a posteriori} probability consists of the product of three terms: a term which is a function of the observation \( r_n \), an \textit{a priori} probability \( \pi_n \) and a term \( z_k \) called the extrinsic probability.

Now considering LLRs, the soft output of the decoder can be written as the sum of the three terms:

\[
\frac{2r}{\sigma^2} + \log \frac{\pi_0}{\pi_1} + z_k
\]

where the extrinsic information \( z_k \) is now equal to the log of a quotient:

\[
z_k = \frac{\sum_{c_l \in C} \prod_{n=1}^{K} p(r_n | d_n^l) \pi_{d_n^l} \prod_{n=K+1}^{N} p(r_n | c_n)}{\sum_{c_l \in C} \prod_{n=1}^{K} p(r_n | d_n^l) \pi_{d_n^l} \prod_{n=K+1}^{N} p(r_n | c_n)}
\]

If the trellis diagram of the encoder is of reasonable complexity, which is the case for convolutional codes of moderate constraint length and for certain block codes, then calculating the \textit{a posteriori} probability can be done by the BCJR algorithm. With this forward/backward type of algorithm, the \textit{a posteriori} probability is evaluated from two probabilities, \( \alpha_j^k(m) \) and \( \beta_j^k(m) \), computed recursively for each instant \( k \) and for each state \( S_k = m \) of the encoder, from the observation \( R^N_k \) and the \textit{a priori} probabilities \( \pi_{d_n^l} \):

\[
\text{APP}_i(d_k) = \sum_m \alpha_i^k(m) \beta_i^k(m)
\]

with:

\[
\alpha_i^k(m) = \frac{\text{Pr}\{d_k = i, S_k = m | R^k_i\}}{\text{Pr}\{R^k_i\}}
\]

\[
\beta_i^k(m) = \frac{\text{Pr}\{R^k_i | S_k = m\}}{\text{Pr}\{R^k_i\}}
\]

In general, the \textit{a priori} probabilities are not known, but they can be estimated from extrinsic information.

The BCJR (APP, MAP) algorithm being relatively complex to implement, several sub-optimal algorithms have been developed to calculate a value close to the \textit{a posteriori} probability or to the LLR. Among these algorithms, the "Max-Log-MAP" proposed by Robertson et al. [19]. To calculate the two probabilities \( \alpha_i^k(m) \) and \( \beta_i^k(m) \), it is necessary to evaluate the logarithm of a sum of exponentials, so the "Max-Log-MAP" algorithm makes the following approximation:

\[
\log(e^{\delta_1} + e^{\delta_2}) = \max(\delta_1, \delta_2) + f_3(\delta_1 - \delta_2)
\]

where \( f_3 \) is a look-up table. With this approximation, the logarithm and exponential functions in the calculation of probabilities \( \alpha_i^k(m) \) and \( \beta_i^k(m) \) disappear, which greatly simplifies the software or hardware implementation without significantly degrading the performance of the decoder (see also [20]). Other algorithms generally classified in the literature as SOVAs (see part I) also enable us to obtain the LLR of a bit to a good approximation.

Iterative or turbo decoder

To present the iterative decoder, let us consider the self-concatenated encoder of rate 1/3, shown in Fig. 5. The C codewords are transmitted by the encoder on a binary-input channel with additive white Gaussian noise. This channel then delivers at each time \( k \) a sample \( r_k \) given by (6).

The samples \( r_k \) are first stored in pairs in a memory of size \( 2K \), similar to that used in the encoder. Each pair is composed of two samples, one relative to the symbol \( d_n \), and the other to the corresponding redundancy (the samples relative to the symbols \( d_n \) are thus stored twice, which of course is not necessary in practical implementations).

The turbo decoder uses a single SISO decoder if the transmission rate is not too high, that is, if it is possible to operate the SISO decoder at a much higher frequency than the transmission rate of the symbols. If this is not the case, each half-iteration of the turbo decoding is done by a dedicated SISO decoder (see Part I). In that case, the delay introduced by the decoding operation is proportional to the number of half-iterations.

On the first iteration, only the observation \( R^{2K} \) composed of \( 2K \) pairs of samples, is available for processing, since the probabilities \( \pi_{d_n^l} \) are unknown (they are arbitrarily preset to 1/2). The decoder first evaluates all the probabilities \( \beta_i^k(m) \) from the observation \( R^{2K} \) by backward processing, then the
probabilities $\alpha_i^k(m)$ for $1 \leq k \leq K$ from the observation $\mathbf{R}_i^k$ by forward processing. An initial extrinsic piece of information denoted $z_i^1$ is then determined for each bit $d_i$. This information is used as the a priori probability of the bit $d_i$ for the second half of the block.

Then the $\alpha_i^k(m)$ calculations for $K + 1 \leq k \leq 2K$ can continue by using the observation $\mathbf{R}_i^{2K}$ as well as extrinsic information $z_i^1$. After calculating each probability $\alpha_i^k(m)$, the decoder again supplies for each $d_i$ another piece of extrinsic information denoted $z_i^2$.

For the following iterations, the probabilities $\beta_i^k(m), K + 1 \leq k \leq 2K$, are calculated from the observation $\mathbf{R}_i^{2K}$ and from the extrinsic information $z_i^2$ of the previous iteration. These probabilities allow a new set of extrinsic probabilities $z_i^2$ to be determined provided that the probabilities $\alpha_i^k(m)$ of the previous iteration have been stored. The computation of the probabilities $\beta_i^k(m), 1 \leq k \leq K$, continues by using the extrinsic probabilities $z_i^2$ of the current iteration, which simultaneously enables the extrinsic information $z_i^2$ to be updated. The process is continued until a hard decision is made on the information bits, by comparing the a posteriori probabilities or the LLRs to thresholds at 1/2 or zero, respectively.

**Performance of turbo codes**

The performance of turbo codes today approaches within about 0.35 dB the theoretical limit introduced by Shannon, for a BER of $10^{-5}$. Figure 6 gives an example of a code offering such performance [21]. Of course, this result is obtained with a large number of iterations (15 to 20, depending on the encoder used), a BCJR decoder and a sufficiently large interleaving matrix (typically $K > 4000$).

Should this gap of 0.35 dB be attributed to the encoder or to the iterative decoding method? Can we hope to reduce this gap or, even more important, reduce the number of decoding iterations for a given performance [21,22]? The principle of turbo decoding relies on the diversity effect of extrinsic information. In order for this effect to fully play its part, it is necessary for the noise affecting extrinsic information (which evolves during processing) to remain uncorrelated with the noise of the channel. This is obviously not possible since the graph associated with a composite code of finite size has finite cycles [23,24]. This is a possible (partial) explanation for suboptimal performance.

**Conclusion**

Iterative processing is commonly used to find solutions to complex problems in many scientific domains: electronics, electromagnetism, mechanics, etc. This type of processing is much less widespread in digital communications. Turbo codes illustrate well the advantage to be gained from iterative processing using all the data available. The turbo principle can be exploited at various levels of the communication chain to improve its performance [25-27].


Many other references can be found at web addresses http://www.ee.virginia.edu/ee/valenti/references.html, thanks to Matt Valenti and http://www.ee.virginia.edu/research/CCSP/turbo_codes/tcodes-bib/tcodes.bib thanks to Eric K. Hall.

About the Authors
Claude Berrou (M’84) was born in Penmarc’h, Bretagne, France in 1951. He received the degree of “Ingénieur” in electronics from the “Institut National Polytechnique” of Grenoble, France, in 1975.

In 1978, he joined the “Ecole Nationale Supérieure des Télécommunications de Bretagne” (French Ministry of Telecommunications), Brest, France.
where he is currently Professor of Electronics Engineering (semiconductor devices, integrated circuits, digital processing, etc.). His research focuses on joint algorithms and VLSI implementations for digital communications, especially error-correcting codes and synchronization techniques. The strong analogy between the Boltzmann’s and the Shannon’s statistical approaches makes his two professional interests, physics and coding, closer than it would appear at first glance.

In 1997, Prof. Berrou was awarded the Stephen O. Rice Prize (IEEE ComSoc) with Alain Glavieux, and the “Médaille Ampère” (SEE Traitement du signal).

Alain Glavieux was born in Paris, France, in 1949. He received the engineering degree from the École Nationale Supérieure des Télécommunications, Paris, in 1978.

He joined the École Nationale Supérieure des Télécommunications de Bretagne, Brest, France, in 1979, where he is a Professor of Electrical Engineering and head of the Signal and Communications Department. His current research interests include channel coding, communications over fading channels, underwater communications, and digital modulation.

In 1997, Prof. Glavieux was awarded the Stephen O. Rice Prize (IEEE ComSoc) with Claude Berrou.

**Historian’s Column**

Continued from page 11

edgment letters to the authors about the existence of this board, was that, if there was such a paragraph, it should be sent only to U.S. authors, since it was “unlikely” that non-U.S. authors would submit their papers to the “Agency.” It should further clearly state that the publication decision would not be influenced by the authors’ actions either way and that the Society neither encouraged nor discouraged cooperation with the “Agency.” For the record, such a paragraph was never included.

Third, its reaction to the question of the report whether the Society would consider sending the papers directly to the “Agency” either with or without the authors’ consent was that such an action would be a gross violation of the confidential nature of the submission process and a violation of the trust placed by the authors in the Society.

Eventually, the furor over this debate subsided (a more accurate term might be “fizzled”) and life returned gradually to its normal state. And the “Agency” continued its work in silence.

**NEW BOOKS**

Raymond Yeung
**Digital Communications,**

Contents: Introduction & background; Data transmission fundamentals; Base-band data pulse shaping; Sources of channel degradation; Bandpass Digital Modulation–basic techniques; Advanced multi-symbol bandpass modulation systems; Coding theory & practice.

**Synchronization Techniques for Digital Receivers,**

Contents: Introduction; Principles; Methods and performance limits; Carrier frequency recovery with linear modulations; Carrier frequency recovery with CPM modulations; Carrier phase recovery with linear modulation; Carrier phase recovery with CPM modulation; Timing recovery in baseband transmission; Timing recovery with linear modulations; Time recovery with CPM modulations; Index.

**Mobile Multimedia Communications,**

Contents: Networks (12 chapters, including): Limits and challenges for wireless ATM, An active network architecture for ATM WANs, Cellular communication systems with voice and background data, Fast handoffs using virtual paths in mobile ATM networks, Comparison of multicast flow control algorithms over combined wireless/wired networks, Protocols (8 chapters, including): End-to-end programmability for QoS controlled mobility in ATM networks and their wireless extension, Loss profiles at the link layer, security for a connectionless peer-to-peer link; Media Access (7 chapters, including): Adaptive packet reservation multiple access (A-PRMA) for broadband wireless ATM, Performance of PBMA with contiguous slot assignment for wireless multimedia networks, Protocol for wireless DS-CDMA network carrying multirate traffic; Signal Processing (9 chapters): Design experience with an integrated testbed for wireless multimedia computing, Impact of antenna choices on the reliability of mobile broadband transmission at millimeter-wave frequencies, content-based video transmission over wireless channels, Video coding with adaptive block update; Index.

**Satellite Communications Systems: Systems, Techniques and Technology,**

Contents: Introduction; Link analysis; Transmission techniques for a satellite channel; Multiple access; Multibeam satellite networks; regenerative satellite networks; Earth stations; The communication payload; Organization of telecommunications platforms; Satellite installation and