The Application of Error Control to Communications

Elwyn R. Berlekamp
Robert E. Peile
Stephen P. Pope

In any system that handles large amounts of data, uncorrected and undetected errors can degrade performance, response time, and possibly increase the need for intervention by human operators

Error Control is an area of increasing importance in communications. This is partly because the issue of data integrity is becoming increasingly important. There is downward pressure on the allowable error rates for communications and mass storage systems as bandwidths and volumes of data increase. Certain data cannot be wrong; for example, no one can be complacent about the effect of an undetected data error on a weapons control system. More generally, in any system which handles large amounts of data, uncorrected and undetected errors can degrade performance, response time, and possibly increase the need for intervention by human operators.

Just as important as the data integrity issue is the increasing realization that error control is a system design technique that can fundamentally change the trade-offs in a communications system design. To take some examples:

1) In satellite communications, high-integrity, low redundancy coding can reduce the required transmit power, reduce the hardware costs of earth stations, and allow closer orbital spacing of geosynchronous satellites.
2) Relative to uncoded modulation, Trellis Coded Modulation allows more data to be transmitted over a limited bandwidth.
3) In situations where data is transmitted on an auxiliary carrier over a preexisting channel, such as data-over-voice modems and subcarrier data transmission within existing broadcast FM and TV signals, coding results in increased channel utilization without altering the existing mode in which the channel is used.

One aspect of coding is unchanged: coding can change data quality from problematic to acceptable. However, the above examples illustrate a new aspect; if a communications system has no problem with data quality, the designers ought to review the design, and consider discarding or downgrading the most expensive and troublesome elements while using use error control techniques to overcome the resulting loss in performance. (A similar strategy also applies to mass memory applications. In these applications, the pay-off is increased storage density which provides both increased capacity and increased throughput.)

From this position, sophisticated coding is increasingly able to offer a unique competitive edge to many diverse areas.

The opening statement that "error control is an area of increasing importance to communications" is also a statement on the history of error control. In fact, the history of error control is very rich and colorful [29]. Coding theory effectively started in 1948 with the appearance of Shannon's classic paper [1]. Early activity was intense and rapidly split between Information Theory, that is what was theoretically possible, and Coding Theory, that is how coding gains could be achieved.

In essence, Shannon's paper proved that a stationary channel could be made arbitrarily reliable given that a fixed fraction of the channel was used for redundancy. Conversely, he showed that, if the fixed fraction was not

0163-6804/87/0004-0044$01.00 © 1987 IEEE

used, reliable performance was not possible. This raised several immediate questions:

1) How could this theoretical performance be translated into practical benefits?
2) What expectations did Shannon's somewhat subtle result arouse?

These questions are discussed below.

Realizing Theoretical Performance

In relation to the first point, it soon became apparent that the practical problem of achieving anything like the performance promised by Shannon was extremely difficult. Moreover, determining the performance limit hinged upon a complete knowledge of the noise statistics on the channel. The philosophy of the early Coding Theorists was roughly as follows: Given that we are informed of, or are allowed to study and classify, the exact noise statistics of your channel, we can prove that your present methods are hopeless and superb performance is possible. However, we do not know how to obtain this performance but we are working on techniques that obtain some small fraction of this performance.

It is not hard to see that the above philosophy is difficult to pursue in a nonresearch environment. In fact, there has been a history of skepticism toward coding from the very start [2]. In terms of the practical application of coding, the Shannon philosophy has fallen out of favor. The real problem is that the noise distribution on a channel is very rarely known and might be impossible to define. For example, the authors recently looked at the noise afflicting data transmitted on a sub-carrier within the bandwidth of an FM station. The noise was primarily related to program content; Pat Benatar was 4.2 times noisier than Count Basie, drum solos excepted. Clearly any absolute definition of noise would involve predictions of trends in popular music. Fortunately, this is not necessary.

The alternative approach (which we credit to Jacobs and Viterbi) was to take a particular code, analyze it under one or more simple types of noise, and present the results in comparison to those obtained with uncoded data communications. Further, in this approach, other codes may be analyzed, shown to be inferior and presented as a "strawmen." The real impact of this approach is that, by offering a tangible product, it throws questions about the appropriateness of a code back to the communications system designer and his knowledge of the channel.

The success of this approach led to the first large scale application of Forward Error Correction to communication, the use of the (2,1) K=7 convolutional code on satellite communications. This application is discussed in the section titled "half-rate coding against predominantly Gaussian noise." One result of this approach is a growing consensus as to which codes are of practical importance, and a growing awareness of the pros and cons of the various classes of codes. To expand on this last point, there are literally hundreds of different codes and decoding algorithms. It is probably possible to take any one of these codes and devise noise conditions and constraints that will make this code optimal. However, these conditions and constraints will be extremely farfetched in the majority of cases. There is a much smaller class of codes that are applicable to a large number of practical problems and a still smaller number of codes that can meet other practical constraints, such as high-speed and/or low complexity implementation.

Expectations of Coding

In relation to the second question, the early expectations of coding theoreticians were high. It was widely held that, if certain problems could be solved, a major revolution in communications would ensue. The problems were seen as threefold:

1) To find good codes.
2) To find decoding algorithms for the codes.
3) To find ways of implementing the decoding algorithms.

In fact, these problems were solved but the revolution did not happen. This was largely for practical reasons. Communications has undergone several revolutions since 1948, mostly of a much less mathematical and more direct nature. This had led coding to be declared to be of no practical importance on innumerable occasions. (Incredibly, entire workshops have been held to promulgate this negative conclusion [3]!). In spite of such declarations, practical acceptance of coding is probably higher now than at any previous time. What has changed?

As was pointed out in the introduction, error control can offer significant gains in systems where other components would be expensive to upgrade. If you accept that a new technology offers a period of high gains followed by diminishing returns, many elements of a communications chain have improved enormously in the last twenty years and are now in the diminishing return phase. Coding is now at the stage of being able to offer an attractive alternative to improving performance.

Perhaps this transformation is best illustrated by the evolution of Reed-Solomon (RS) codes. The existence of the codes was published in 1960 [4]. Their status for the next decade was almost totally academic. In 1966 G.D. Forney [5] established that RS codes could be concatenated to supply extremely good performance but, at the time, the existence and implementation of decoding algorithms were daunting problems. Discovery of good decoding algorithms was a gradual task, solved by the efforts of several people in the mid to late 60s, a breakthrough occurring with the publication of Berlekamp's algorithm [6]. In fact, the search for more efficient decoding algorithms has continued ever since; the definition of "efficient" depending upon the technology being used for implementation.

Reed-Solomon codes were still regarded as of academic interest at the start of the 70s. (The late 60s and early 70s mark the lowest point in the history of practical forward error correction; expectations were low. Conversely, the theoretical side of coding was in good shape). By 1982, this was not so. RS codes were incorporated in a Deep-Space standard [7], used in the JTIDS system and commercially available in popular Compact Disc systems. What were the causes of this turnabout?

Partly, it was the explosion in digital circuitry. RS codes are highly suited to digital implementation and, as the capabilities of digital electronics grew, their implementation became possible. It was also due to their performance. In many ways RS codes have complementary properties to convolutional codes. If a system designer has doubts that his channel will exhibit noise conditions suited to convolutional codes, there is strong likelihood that an RS code will function under these conditions. (See the section titled "Half-rate Coding for Predominantly Gawsian Noise.")

Note how this fits in with the changing strategy described in "Realizing Theoretical Performance." The communications designer is being offered a limited number of coding options and, depending upon the channel noise conditions and other constraints, the "best" option will change. It is the purpose of this article to review the properties of various error control schemes in order to clarify the properties of the available techniques.

Error Control Techniques

There are several divisions between types of error control. The first major division is between Automatic Request for Retransmission (ARQ) and Forward Error Correction (FEC). In the communications context, both techniques add redundancy to data prior to transmission in order to reduce the effect of errors that occur during transmission. However, the philosophy is very different.

Forward Error Correction utilizes redundancy so that a decoder can correct the errors at the receiver. There does not need to be a return path. ARQ utilizes redundancy to detect errors and, upon detection, to request a repeat transmission. A return path is necessary.

Adaptive techniques such as fault tolerance could be regarded as another form of error control. For example, in a network it is possible to avoid errors by routing traffic around damaged parts of the network (adaptive routing).

The remainder of this section introduces the fundamental properties of ARQ and FEC techniques. The section titled "Comparison of Techniques" compares techniques and their properties in greater depth. (Reflecting the author's bias, the emphasis is on Forward Error Correction.)

It should be apparent that hybrid strategies involving combinations of the above techniques are possible. These are discussed in the section "ARQ and Hybrid ARQ/FEC Strategies." The section titled "Applications" gives some practical applications of FEC techniques.

Forward Error Correction Techniques

The strategy of Forward Error Correction is to get it right the first time.

The underlying tenet of FEC (in a communications system) is to take user data, add some redundancy, and transmit both the user data and the redundant data. At the receiver, the possibly corrupted signal is processed by the decoder which utilizes the redundancy to extract correct data. An analogy may be made to a person speaking slowly and repetitiously over a noisy phone line, adding more redundancy for the listener to process into the correct message.

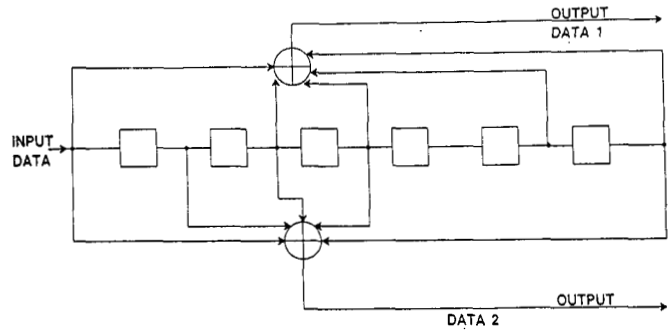

Fig. 1. A (2,1) K=7 Odenwalter Encoder.

Note also that FEC does not need a return path; in our analogy, the speaker could be dictating into an answering machine.

Forward error correction codes come in many diverse forms with many diverse properties. One immediate division is into *convolutional codes* and *block codes.* These are described in outline in the next two sections.

convolutional codes; the (2,1) K=7 *Odenwalter code*

Figure 1 shows a flow chart for an encoder for the (2,1) $K=7$ convolutional code generally used in commercial satellite applications. The encoder processes a continuous stream of data. One bit of user data enters the register. By adding certain stages of the register, two bits of data are obtained and transmitted. Note that the value of the output bits depends on seven user data bits; this accounts for the $K=7$ notation. K is referred to as the constraint length. Note also that a particular bit of user data affects the output for seven clock cycles; the encoder is adding time diversity.

Decoding FEC codes is normally more complex than encoding. For this code, decoding is most often done by the Viterbi Algorithm [8], although other alternatives exist [9]. Whichever method is used, it should be clear that the decoder has to know the history of the decoded stream, that is the values held in the shift register, before being able to decode a particular bit. Furthermore, it has to look at the subsequent history of the stream to examine the total influence of that bit on the transmitted stream.

This suggests that with convolutional codes:

1) If the decoder loses or makes a mistake in the history of the stream, errors will propagate. Such events can be caused by a fading or burst interference channel.
2) The decoder has to perform a large number of operations per decoded bit.

These questions are examined further in "Code Comparison for Gaussian Noise Environments."

block codes; Reed-Solomon codes

Block codes do not process data in the same continuous fashion as convolutional codes. The user data is split

into discrete blocks of data and each block is independently processed by an encoding algorithm to add redundancy and produce a longer block. The decoder works on a similar basis; each block is individually processed. Note that the decoder has to be informed of the block boundaries; synchronization is normally more of a problem for block codes than convolutional.

Block codes are very diverse. The encoding and decoding algorithms often employ many sophisticated finite algebraic concepts. Although this makes block coding into a specialist area, the use of finite algebra is very suited to implementation in digital electronics, allowing for high-speed operation. (Of course, this does not imply every digital implementation is well-suited; the authors have observed different implementations of the same code that differ by a factor of 8-10 in IC count!)

This article intends to emphasize practical uses of Reed-Solomon codes. There are many reasons for the practical importance of RS codes, some of which are described in "Code Comparison for Gaussian Noise Environments." This section confines itself to describing the parameters and correction power of the code.

RS codes operate on multi-bit symbols rather than individual bits. Consider a RS(64,40) code on 6-bit symbols. The encoder groups the user data into blocks of 240 bits. These blocks are treated as 40 symbols where each symbol has 6 bits.

An encoding algorithm (see [6] for details) expands these 40 symbols to 64 symbols. In a systematic RS encoder this is achieved by appending 24 redundant symbols (non-systematic RS encoders exist which scramble and expand the data; there are many incompatible RS codes with the same parameters).

In general, RS codes exist on b bit symbols for every value of b. RS(n,k) codes on b bit symbols exist for all n and k for which:

$$0<k<n<2^b+2$$

Clearly, there is a lot of choice.

(A popular value for b is 8; in this case, the symbols are bytes. 8 bit RS codes are extremely powerful.)

It remains to consider the correction power of RS codes. The general formula is that, if there are $r=n-k$ redundant symbols, t symbol errors can be corrected in a codeword provided that $2t$ does not exceed r. In the RS(64,40) example, every pattern of up to 12 symbol errors is correctable.

A mnemonic is often used for the above formula. The decoder has r redundant symbols "to spend." The decoder has to "spend" one redundant symbol to locate an error, and another redundant symbol to find the correct value of the symbol in that location. Given that the total expense can not exceed r, the formula results.

This mnemonic is also useful in introducing erasure decoding. Suppose some external agent were to inform the decoder where some of the errors were located. In this case, the decoder would not have to "spend" symbols locating these errors. More technically, if the decoder is instructed that s symbols are unreliable (the symbols are referred to as "erased") and, in addition, there are t errors that the decoder is not informed about, the correct data can be extracted provided that $2t+s\leq r$.

Even on a channel with random errors, it is not immediately obvious how these formulas translate into performance. To give some appreciation, Fig. 2 shows the performance of RS(64,k) codes against random digital errors for all values of k divizable by 4. The performance is for decoding of errors only (no erasures).

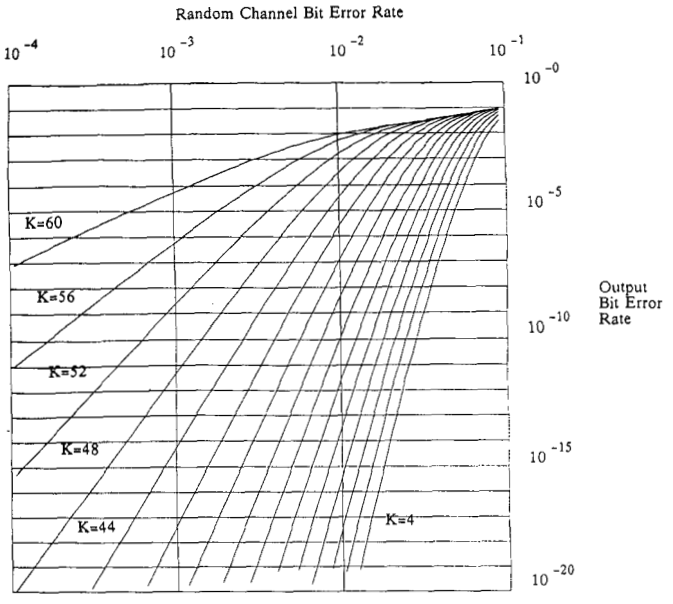

Fig. 2. RS(64,k) Random Digital Error Performance.

Automatic Request for retransmission

There are many different types of ARQ. ARQ techniques include, amongst several others, "Selective-Repeat" and "Go-back N" systems. In addition, there is a logical split between positive and negative acknowledgment systems. The differences are conceptually not difficult.

positive and negative acknowledgment

In a negative acknowledgment system, the data source re-transmits data only upon request; no news is good news. In a positive acknowledgement system, the data source expects confirmation from the data sink on the correct receipt of every block. In a positive acknowledgement system, the throughput can be low, particularly if the round-trip delay between the data source and the data sink is long, that is, over a satellite link. The advantage of positive acknowledgement is, of course, superior data security. In practice, the majority of commercial ARQ systems default to negative acknowledgement with positive acknowledgement as an option.

"go back N" and selective-repeat protocols

In a "Go back N" system, the ARQ is normally negative and the data sink requests that all of the last N blocks are repeated whenever a block is received in error (popular values of N are 4 or 6). A "Go back 4" system can be found in the EUROCOM D/1 tactical area system trunk-group protocols.

The advantage of a "Go back N" system is that the blocks do not have to be individually labeled and that the algorithms are accordingly simpler than a comparable

selective repeat scheme. In fact, the lack of numbering leads to less overhead and shorter blocks which, in turn, means that the performance in random noise can be slightly superior. However, there are disadvantages with "Go back N" systems. In any "Go back N" system, in order to make sure the erroneous block is repeated, the maximum round-trip delay on the link and, hence, the maximum range, is limited. The delay can not exceed the time it takes to transmit N blocks. For satellite links, this restriction is a major problem. Another criticism of "Go Back N" systems is that re-transmissions send more data than necessary.

In a selective repeat system, the ARQ is normally negative and the data-sink asks for specific blocks to be repeated. The advantage of Selective-Repeat Protocols is that only blocks with errors are repeated. In many noise conditions, these ARQ protocols are best in achieving good throughput. In a Selective-Repeat protocol, blocks have to be individually labeled. This indicates that the protocol overhead is either higher than a comparable "Go Back N" system or that the blocks are longer. While long blocks solve many problems, they are more likely to contain errors than a shorter block length. This presents a trade-off between minimizing the loss of throughput due to protocol overhead (long blocks) and minimizing the loss of throughput due to repeats (short blocks). Selective-Repeat protocols are more complex than "Go Back N" protocols but they have much less restriction as to range. However, some limitations still exist. The blocks have to be numbered individually and the length of the counter upper bounds the maximum range. In practice, the range is more likely to be determined by the amount of memory that the data source has available to store transmitted blocks. Note that as the speed of satellite links increases, the time delay of a satellite link is translated into an increased number of blocks for the data source to store.

Comparison of techniques

In this section we examine and cross-compare the various techniques that have been introduced in the previous section. In accordance with the major divisions, the comparison falls into two parts.

1) A comparison of Convolutional codes to block codes. This is discussed in terms of a predominantly Gaussian Noise channel.
2) A discussion of the relative merits of FEC, ARQ, and hybrid schemes.

half-rate coding for predominantly Gaussian noise

The use of half-rate codes to combat Gaussian noise has a long history in satellite technology. The impact of the (2,1) K=7 convolutional code described earlier has been profound. In pacticular, the success of this code has been an important element in changing industry's perspective of coding from being a mathematical curiosity to being an important and practical element in many communication systems.

The nature of satellite communications appears to be changing, both in the amount of traffic (increasing) and type of traffic (from digitized voice to intermachine data). Both of these changes present more challenging requirements on the technology. For example, whole new families of modulation schemes have appeared to make better use of the available bandwidth and transponder power [10],[11]. On the coding level the change has been away from conditions for which the Odenwalter code is suited and towards conditions for which other coding strategies are more apt. The reasons for this change in code suitability are outlined in this section.

Convolutional Codes

This section confines itself to discussing the (2,1) K=7 code and decoders using the Viterbi Algorithm. Gaussian noise performance curves for this code are shown in Fig. 3.

The code is extremely good when all of the following conditions exist:

1) The noise is white and Gaussian.
2) The demodulator provides reliable *soft decision* information (probablistic information on the likelihood of a received symbol being a 0 or 1).
3) The output data (user data) only needs a low level of integrity such as an output BER of 10^{-3} to 10^{-7}.
4) The transmission speed is low enough to allow the decoder to perform a relatively large number of operations per bit.

These conditions, and what happens when they are contravened, are now discussed in more detail.

Gaussian noise—The Viterbi Algorithm is very good at processing soft decision data into an estimate of the transmitted data. However, the complexity of the Viterbi Algorithm confines its application to codes with a small constraint length. K=7 is appropriate at conventional speeds; K=11 is about as large as feasible even on very slow channels, and even K=5 may be difficult on chan-

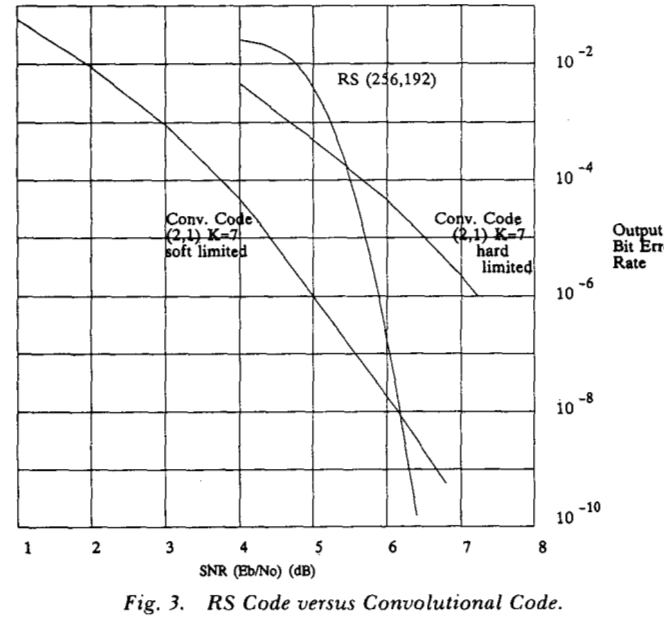

Fig. 3. RS Code versus Convolutional Code.

nels whose throughput is hundreds of millions of bits per second.

This means that the decoder processes the data within a fairly small window before forming a decision as to the transmitted bit. If the data is badly corrupted or the soft decision information is unreliable within that window, the decoder is likely to go wrong and output a burst of errors. Such conditions can exist both for interference channels and for fading channels e.g. non-white Gaussian noise. It is worth noting that the world is getting less white and less Gaussian. Adjacent channel interference, accidental or deliberate interference, antenna misalignment all give conditions not intrinsically suited to convolutional codes.

Reliable soft decision information—For white Gaussian noise, a decoder can be supplied with reliable soft decision data relatively easily. The maximum likelihood metric simplifies to the inner product of the received data and the hypothesised transmitted data [12]. For fading channels and/or high-power interference, this simplification is not valid. For fading channels, the received SNR must appear in the metric. This at the very least complicates the decoder, and also raises questions as to whether the demodulator/AGC can supply the received SNR with adequate precision.

Low output data-integrity—The performance of a convolutional code depends upon the constraint length. Because this length is normally K=7, the performance is limited. It is possible (if not too helpful) to speak of the constraint length of block codes. Typically, this length is very much larger (for example, K=2000). How does this difference appear? The key to appreciating the different performance characteristics of block and convolutional codes lies in splitting the effect of the code into two parts:

1) The code adds diversity which averages the effect of the noise over a number of bits that depend on K.
2) Provided the averaged received noise does not deviate too badly from the expected level of average noise the decoder can use the redundancy to repair the noise damage.

It takes little thought to realize that a very large value of K makes the averaged level of received noise much more predictable. For example, in six dice throws we are not shocked to see two fives appear on the dice. In six hundred throws of the dice, we would be very surprised to see two hundred occurrences of five.

Looking at Fig. 3 we see the characteristic steep slope of the performance curve for the block code versus the more graceful degradation of the convolutional code. Which is "better"? The answer depends on several variables. For digitized voice the convolutional code is better. Such traffic is intelligible in a BER of 10^{-3}. For blocks of machine-oriented control data, the block-coded data is better; BER requirements are typically in the 10^{-10} to 10^{-11} range.

Transmission speed and decoder complexity—Convolutional decoders have a high complexity in terms of decoding operations per output bit. Furthermore, a majority of these operations involve addition and comparison of real numbers (or, at least, digitized approximations to real numbers).

By contrast, block codes have a much lower number of decoding operations per output bit. Moreover, Reed-Solomon decoders normally operate directly on bits and real number arithmetic is avoided.

Equally important is the observation that the convolutional decoder complexity increases as the redundancy decreases. For block codes, the complexity decreases as the redundancy decreases. For high code-rates (such as, 14/15 rate) this greatly favors block codes.

In practical terms, the above considerations limit the speed of convolutional decoders. The authors are aware of paper studies into single un-multiplexed convolutional decoders that can work at 120 Mb/s but are unaware of any existing machines that work beyond 20-30 Mb/s.

Conversely, powerful RS block decoders have been built and delivered that operate at rates above 120 Mb/s. RS decoders that operate at channel rates in excess of 2.0 Gb/s are currently being developed [28].

The speed issue is becoming increasingly important as the switch to multiplexed traffic continues. It is becoming less efficient to protect individual slow traffic channels and more economic to protect the grouped channels at the aggregate rate.

Block Codes

There are many different families of block codes. Of major importance for practical applications is the family of Reed-Solomon codes. The reason for their preeminence is that they can combat combinations of both random and burst errors. They also can have a long block-length, assuring a sharp performance curve. The major drawback with RS codes (for satellite use) is that the present generation of decoders do not make full use of bit-based soft decision information. They can handle erasure-information, an indication of when a multi-bit character is unreliable, but not intermediary levels of confidence.

In fact decoding algorithms have been implemented which make use full-use of soft decision information on the byte level [20], but most presently available decoders do not have this advantage.

Interleaving

The section "Code Comparison for Gaussian Noise Environments" indicates that convolutional codes are weak when it comes to burst noise and that Reed-Solomon codes are superior [15]. A common counterargument is that the codes can be interleaved so that the un-interleaved received data is more or less random and, therefore, more or less optimal for the convolutional decoder. After all, Reed-Solomon codes are often interleaved.

This argument is appealing but fallacious. In a very precise information-theoretic sense, the very worst type of noise is random noise. If the noise has structure, the structure can be exploited. In less high-brow terms, the effect of short interference bursts on a telephone line disturbs reception less than a constant level of white

noise. The reason for this is not profound. The receiver can locate when a burst occurs and make good use of the location information. In an interleaved code, the de-interleaver carefully tries to convert a less damaging type of noise (bursts) into a more damaging type of noise (random). Given that interleaving is wrong on a philosophical level but practically necessary in many applications, a useful design methodology is to start with a code that has considerable burst-error ability and interleave to a much lesser extent.

In more detail, system delay constraints typically preclude interleaving to a depth so great as to make the channel appear memoryless; there is often a nonnegligible probability that a channel noise burst will have length which is a significant fraction of the system delay constraint. In such cases, the proper comparison is not between idealized coder-interleaver systems with infinite interleaving depth, but rather between specific schemes that meet the delay constraint.

When this delay constraint is tens of thousands or hundreds of thousands of bits, and bursts occasionally approach such lengths, then long RS codes with well-designed interleavers enjoy an enormous performance advantage over interleaved convolutional codes. However, when burst length and delay constraints are both only tens or a few hundreds of bits, then long RS codes are precluded. In this case, interleaved convolutional codes with Viterbi decoding may be the only viable solution.

Concatenated Codes

In many applications concatenated codes offer a way of obtaining the best of two worlds. The idea is to use two codes in series, as shown in Fig. 4. Why, it can be asked, use two codes? If one can not work, why use two? An answer can be made on many levels, but for the satellite case, the following summarizes why this approach works well.

The inner decoder processes the incoming data. This decoder uses all available soft decision data to obtain the best performance in Gaussian conditions. However, the inner decoder is normally limited in performance and is vulnerable to interference and fading. The effect of the inner decoder is to clean-up the majority of Gaussian noise and to indicate stretches of data with which it could not cope. The output of the inner decoder is predominantly either correct data or stretches of burst-errors that are indicated as being unreliable. This output then becomes the input to the RS decoder. From the "Comparison of Techniques" section it should be clear that the RS decoder is now being presented with its ideal input

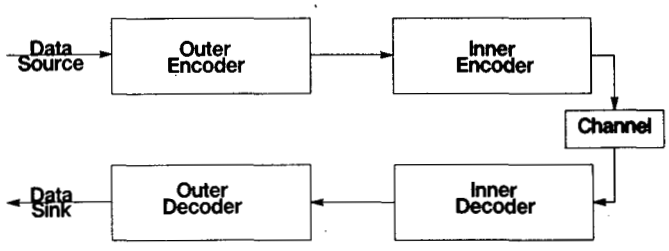

Fig. 4. Concatenated Code Schematic.

that is, mostly short burst errors accompanied by helpful erasure information.

The effect of using the two decoders, each combating its preferred type of noise, results in strong performance. The exponential nature of code performance comes into play. The inner decoder reduces poor quality data to medium quality data and the outer decoder reduces medium quality data to very good quality data. The performance of the sum is much greater than the sum of the performance either part could achieve alone.

However, the above description is deliberately vague about the inner code selection. Should this be block or convolutional? The choice depends on the application, the data rates and the channel conditions.

For very high data rates, the choice is for block codes. An inner convolutional decoder would fail to keep up. For predominantly Gaussian slower channels, the inner convolutional decoder is very attractive. Under these conditions, the convolutional inner code tends to outperform most short block codes of interest (ignoring some types of very low rate spread-spectrum systems). The section "The CCSDS Standard" Concatenated Block and Convolutional Code" discusses the performance obtained by concatenating a low redundancy (8 percent) RS code with the (2,1) K=7 Odenwalter code. If the noise is predominantly not Gaussian, the choice is for an inner block code. Each inner codeword can protect one character of a Reed-Solomon codeword; the effect of a burst can not propagate through the inner decoder and corrupt more characters than necessary. The section "A Concatenated Reed-Solomon Binary Block Code System" discusses the performance obtained by concatenating a short binary block code with an RS code.

Summary of Comparison for Half-Rate Coding

The continuing growth of data communications over satellite presents new challenges and reappraisals to the communications engineer. In particular, the characteristics of convolutional codes are much less suited for the demands of high-speed data needing high integrity than for their traditional application of low-speed digitized voice traffic. The required performance tends to favor Reed-Solomon codes. RS codes exhibit a very sharp improvement of block error-rate with an improvement of channel quality making their use ideal for data. However, RS codes used in isolation fail to make good use of the soft decision data which is both available and reliable in commercial satellite applications. In order to exploit all the available information and to obtain the RS performance at even worse signal-to-noise ratios, concatenated codes can be used to great effect in this application. If either extremely fast data-rates or very little redundancy is available for coding, concatenated codes are less suitable; RS codes alone may be preferable.

comparison of high-rate codes under Gaussian noise conditions

"Half-rate Coding for Predominantly Gaussian Noise" discussed the role that half-rate coding has played and is likely to continue to play in Gaussian noise environments, such as satellite communications.

However, the pressure for bandwidth and the increased performance of modern codecs raises questions about the need for so much redundancy. This section addresses what can be done with a minimum of redundancy, for example, 5-25 percent.

The question falls into two overlapping portions: modulation and coding. The introduction of trellis-coding into commercially-available modems has finally shown that combined modulation and coding provides gains which far outweigh the traditional operational and maintenance advantages of modularity. Trellis coding uses convolutional coding but "buries" the redundancy into a more complex signal constellation. This constellation presents worse performance in isolation, but this is more than compensated for by the coding gain. The net result is that the gain can be used to send more bits over the same bandwidth, seemingly without redundancy. TCM is discussed in [13]. Research is now extending coding further into the communications chain, for example, the decoding/demodulation process is being extended to interact with or include the channel equalization process.

"Half-rate Coding for Predominantly Gaussian Noise" discussed the drawbacks in using convolutional codes alone. Unfortunately, the same criticisms can be applied to trellis coding used alone. The coding is vulnerable to fades and interference bursts. In very-high speed applications, above 20 Mb/s, it will be difficult to implement the decoders/demodulators. In such applications, high-performance, low-redundancy block codes present a cost-effective option used either with simpler modulations or concatenated with trellis codes.

"Half-rate Coding for Predominantly Gaussian Noise" presented the advantages in concatenating RS codes with conventional convolutional codes. The same reasons apply to concatenating with trellis codes. However, just as trellis codes can be regarded as an unconventional form of convolutional codes, the most efficient form of block code to concatenate is probably not a conventional block code. (In fact, the authors have developed a class of block codes that are very closely matched to the noise statistics produced by a trellis decoder.) The remainder of this section discusses what can be achieved with simple modulation schemes and codes that use only a small percentage of the available bandwidth.

If the redundancy is to combat the noise, the noise has to be a correspondingly small percentage of the codeword. In order to ensure that this occurs most of the time, the received noise has to be averaged over a long period of time, reducing the effect of a freak streak of bad luck; the code must have a large block length or constraint length. This immediately rules out convolutional codes. The Viterbi Algorithm is only economic for small constraint lengths. Sequential decoding is often promoted as a way of increasing the constraint length, but sequential algorithms typically have the undesirable property that the variance in the number of decoding operations per bit is infinite [14]. Thus, in order to achieve any reasonable rate of continuous throughput, a sequential decoder may require an average operating speed much faster than the channel, as well as a very large buffer and a very large decoding delay.

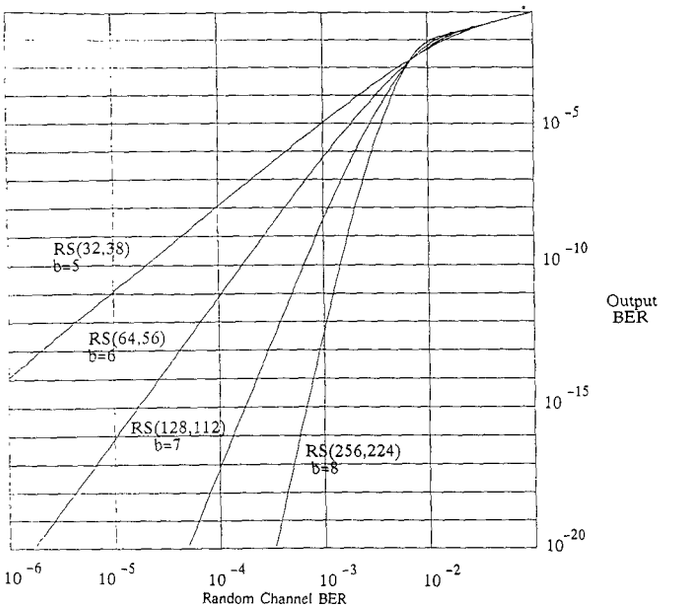

Fig. 5. RS Codes of 7/8-th Rate

In order to meet the long block length requirement, one attractive alternative is to use RS codes. The effect of block length is illustrated by Fig. 5. Figure 5 shows RS codes working at rate 7/8. The first code is a 5-bit code that is block length 31*5=155 bits. The other codes are 6,7, and 8 bit codes of the same rate. Notice that the 8-bit code's (block length 255*8=2040 bits) performance is much more "knife-edged". For data communication this is to be preferred; BER performance in the 10^{-20} region is obtained at much lower SNRs.

There are other reasons why RS codes are suited to this role:

1) RS codes are symbol or character based codes rather than binary codes. For example, the RS(255,223) code takes 223 bytes of data and encodes them into 255 bytes of data. The decoder and encoder logic works with byte-based arithmetic. The decoder processes 255 items of data, not 255*8 items of binary data. This reduces the complexity of the logic as compared to a "true" binary code of the same length. In addition, decoding algorithms exist for RS codes that are extremely efficient at reducing the number of operations per output bit. This allows the increase in block length to be implementable. In fact, the number of operations per decoded bit grows with the number of redundant symbols; low redundancy RS codes are less complex than high redundancy RS codes.
2) RS codes have a very low misdecode rate. If the noise exceeds the amount that the code can correct, the decoder will almost always recognize the impossibility of decoding the data and can flag the data as being corrupt. The error-rate of data that is both undecodable and NOT recognized as such by the decoder is typically 5 orders of magnitude below the overall BER. This property is very valuable for inter-machine data. Furthermore, this bad data signaling can be used in other parts of the communi-

cations chain, for example, for ARQ purposes, for Automatic Power Control purposes and for channel assessment purposes.

3) RS codes work on a block basis. One of the consequences of working with data is that the data often has a natural frame or packet size. For such blocked data, the statistic of how many bits per block are wrong, the BER, is much less important than the probability of getting a block right. This latter quantity can be computed accurately if the block code is designed to synchronize with the frame or packet boundaries. Exactly the same comments apply to the misdecode rate.
4) The symbol structure of RS codes tends to absorb short bursts. The point is that, as far as the decoder is concerned, having several bits in the symbol wrong is no worse than having one bit wrong. If the noise tends to "bunch," as, for example, in multibit baud modulation, the performance can be better than for purely random noise.
5) Finally, RS codes can often be easily accommodate within a multiplexed traffic stream. The codes can be systematic, with the data untouched and check characters appended. This allows for a multiplexed frame where the majority of the channels are dedicated to user data, frame-sync or control channels, and a minority of the channels are dedicated to code check characters. Note that the code delay is divided equally amongst the individual channels. A practical example is discussed in the section titled "Block Codes on a TDMA System."

In summary, high-rate coding against Gaussian noise tends to indicate:

1) Convolutional coding for low-speed, low-integrity applications.
2) Concatenated convolutional and block coding for low-speed, high-integrity applications.
3) Block codes alone for high-speed and high-integrity applications.

Of these three areas, the last two seem to be of increasing interest to satellite communications.

ARQ and Hybrid ARQ/FEC Strategies

ARQ at the link level

Consider first the effect of using ARQ on a link. The throughput of ARQ alone (often called Type 0 ARQ) is very dependent on the channel conditions. Figure 6 shows the throughput of 1000-bit blocks of ARQ-ed data when faced with Gaussian noise. The throughput collapses when the expected number of errors per block becomes sizable.

Periodic interference can hurt ARQ techniques for the same reason. In one example, an airport radar was found to be corrupting data communications on each sweep. This translated to a burst error on a once-per-block basis, giving virtually zero throughput.

One adaptive technique is to shorten the ARQ block length as the noise worsens. This does help the throughput, but there are drawbacks. Firstly, the amount

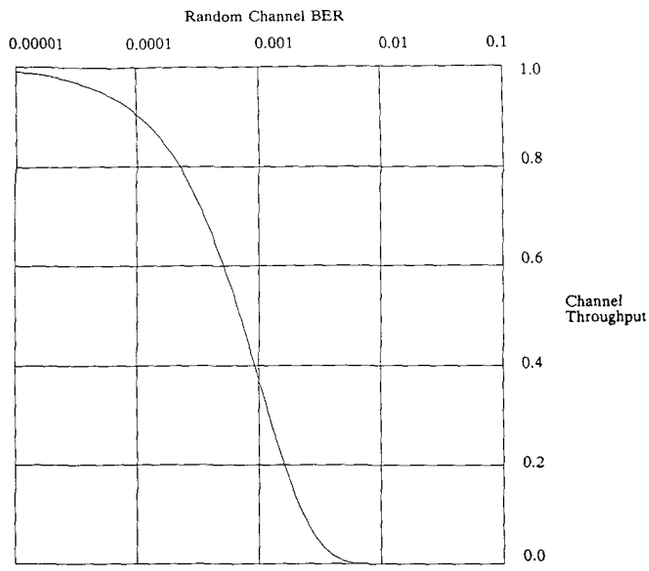

Fig. 6. ARQ Throughput for 1000-Bit Blocks.

of protocol overhead per block is normally fixed. Hence, as the blocks are shortened, less data is transferred. Secondly, the technique only helps within fairly limited bounds against random errors. Even with optimally sized blocks, the low throughput of ARQ under non-benign conditions remains a major disadvantage.

In summary, Type 0 ARQ is probably most applicable to channels which are mostly error-free but subject to infrequent bursts of interference of limited duration.

The most important advantage of ARQ is that the delivered data from ARQ has predictable quality; the greatest disadvantage is that throughput is dependent on channel conditions. FEC behaves in a complimentary fashion: throughput is constant, with data quality depending on the channel. This immediately suggests combinations of FEC and ARQ.

One common hybrid FEC/ARQ scheme is called Type 1 ARQ. In Type 1 schemes, the initial aim is to correct data using FEC. If this fails, the error-correcting decoder can be designed to have a high degree of residual error-detection, and this detection can be used to trigger an ARQ mechanism.

Although this sounds logical, the effectiveness of Type 1 depends, as ever, on the type of channel noise. If the link is uniformly noisy, there are drawbacks with Type 1. Figure 5 shows the performance of several RS codes. Notice how steep the curves are. The addition of ARQ under these random error conditions is of mainly psychological benefit. If the code is working, the ARQ is redundant. If the code is not working, the ARQ will not help as the possibility of getting a correct repeat is remote. The combined FEC/ARQ is only helpful in the narrow descent region. For levels of noise that slowly vary in intensity between reasonable bounds, adaptive FEC techniques can be more apt than Type 1 FEC/ARQ, that is the protocols adapt the FEC code, redundancy and interleaving to the prevailing noise conditions. Conversely, if the link is mostly uniformly noisy but prone to infrequent onsets of severe bursts of interference or

fading, Type 1 FEC/ARQ can be very effective. An example of a Type 1 FEC/ARQ scheme can be found in [15].

One objection to Type 1 schemes is that the FEC redundancy is present even when the noise is not. This, amongst other reasons, has led to the design of Type 2 ARQ/FEC schemes [16]. There are many different types of Type 2 ARQ but the predominant characteristic is to use ARQ first and FEC second. This is best explained by example.

In one Type 2 system [16] data is sent out without FEC protection but with redundancy attached for error-detection and ARQ purposes. Suppose that n bytes of data are sent in a packet. If the data is received correctly, then there is no problem. If errors are detected, a request is sent back to the data source. The data source does not repeat the data. Instead the source uses a $(2n,n)$ systematic code to obtain n check bytes and transmits the check bytes, again with error-detection attached.

The data sink now has a choice of options. If the check bytes were received correctly, an inverse permutation is applied to extract the correct data. If the receiver detects errors in the check bytes, the receiver has an erroneous block of message bytes and an erroneous block of check bytes. The two blocks are then passed to a $(2n,n)$ error-correcting decoder which attempts to extract the correct data. If this succeeds, the process is over. If not, another request is sent to the data source and the process is repeated, that is, the data bytes or a new set of n check bytes may be repeated.

network level

The application of ARQ to networks (packet-switched networks in particular) is extensive.

The premise of packet-switched networks is that there is much more bandwidth in a network than in any single link. Data can be sent quickly if it is packetized and the packets are transmitted over several routes in parallel.

This approach fits in well with selective repeat ARQ; if one or more of the packets hit a bad or noisy link, then this packet can be repeated and, with high probability, the repeat will be sent on a good route.

Currently there is considerable work and interest in adding adaptive routing techniques to packet-switched nets, i.e. the network isolates faulty links and does not route data over these links. Note that the data communications equipment at each end of a faulty link can continue to send test data over the faulty link, monitoring for any improvement.

The motive for adaptive routing is straightforward; if a link is not working why bother to use it? The only point of debate is about the definition of when a link is "working" or not. Suppose no link is working? Can we make more links work with FEC and ARQ? How quickly can we route around new centers of damage and bring improved links back?

These points are not academic; the success of packet-switched technology in local area networks has led to their use over military radio networks where there is a threat of Electronic Counter Measures (ECM).

The implications of ECM are now discussed in greater detail.

Obviously, adaptive routing strategies fail if every single link is not "working". Furthermore, Fig. 6 indicates that Gaussian noise can render Type 0 ARQ strategies ineffective on the link level. Furthermore, the levels of Gaussian noise required to do so would be considered mild in the ECM radio environment: an obvious jamming strategy is to corrupt every link with relatively low-levels of noise. This can be protected against by adding FEC, as well as or in place of ARQ, to each link, for example, Type 1 FEC/ARQ.

Assuming these measures have been taken, this presents the jammer with three choices. He can:

1) Give up (the ideal solution from the communicator's position).
2) Up the stakes and jam all the channels at much higher power levels.
3) Concentrate his resources on a few of the links.

In fact the latter might be forced on him by geographical considerations.

In either of the last two cases, the most likely result is that some of the links will be operative and some will not (the dichotomy being strengthened by the knife-edged performance of good codes, see Fig. 5). In these circumstances, an adaptive routing strategy can be effective.

However, adaptive routing does not solve every problem. When an ECM attack starts, the re-routing must take some finite time to notice, diagnose and re-route traffic around areas of damage. This time can not be made too short; it is not desired to close links at the slightest defect. Unfortunately, in the time period immediately after an attack the following is likely to be true of the network:

1) It is most necessary to transmit command and control data quickly and efficiently.
2) It is in this time-period that the delay caused by the attack is most severe.

To quantify these points, Curve 1 of Fig. 7 shows the normalized time delay that results from an ECM attack in

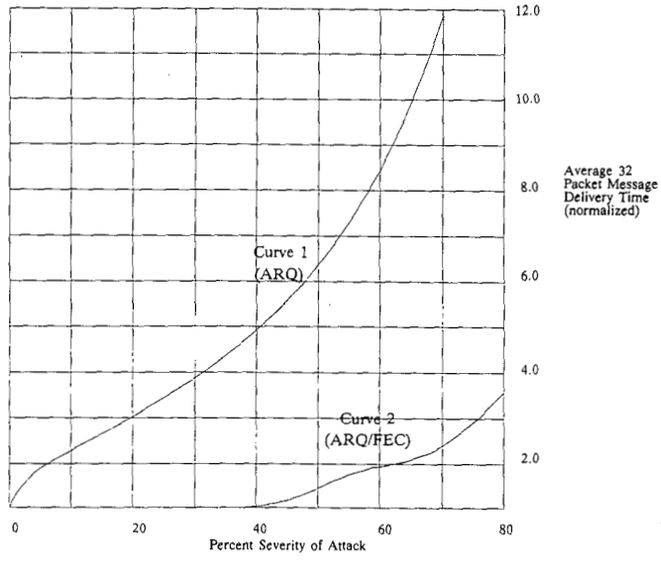

Fig. 7. Average Time-Delay in an Attacked Network.

the absence of re-routing. The x-axis shows the severity of the attack in terms of the percentage of non-working routes from A to B. The y-axis shows the average delay in sending 32 packages of data. In the absence of damage the time is normalized to 1 unit. Even with adaptive rerouting, in the period immediately after the attack, Fig. 7 could still be valid due to the time taken to re-route.

Another concern with adaptive re-routing is spoofing. The ECM can jam some percentage of the channels until the re-routing closes them down. The jammer can then jam some other set of channels, safe in the knowledge that there will be a time lag before the closed channels can be re-opened.

The above discussion indicates that for a packet-switched network to be survivable against sustained ECM attacks, error-control must be present on several levels. FEC/ARQ is necessary at the link level to make it as difficult as possible for a link to be rendered inoperative. In addition, FEC/ARQ techniques on an end-to-end basis are necessary to protect against the detrimental effect of inoperative links.

Suitable end-to-end ARQ techniques are already supplied by packet-switching and adaptive routing techniques, but suitable end-to-end forward error correction network techniques have received relatively little attention. The authors have developed one example of packet-coding; another example is the "code-combining" method of D. Chase [17].

Curve 2 of Fig. 7 shows the performance of a packet-coded ARQ scheme in transferring 32 packets of data. Note that it is uniformly superior to the uncoded case and that negligible delay is caused if less than 30 percent of the channels are jammed. When this is added to the gains due to coding on the link level (that is, when it is difficult to jam one link, let alone 30 percent), the task of the jammer is seen to be much harder.

The above discussion and examples are designed to show how adaptation offers an effective solution to a wide range of communication problems. The key to adaptive systems is a thorough knowledge of the parameters that should be adapted and the properties of the different techniques that can be deployed. From this point of view, FEC offers a set of properties that are complimentary to the techniques that have been most often used to date. As the demands on communication systems become more severe, the co-design of systems using FEC as an ingredient is expected to bring about a rapprochement of modulation, ARQ, multiplexing, packet-switched and protocol design.

Applications

This section looks at some particular examples where error control has been applied. The examples have been chosen to illustrate some of the points discussed in the previous section.

Block Codes on a TDMA System

A high rate (16/17) Reed-Solomon codec has recently been integrated into a satellite TDMA system [18]. The advantage of using such a high-rate code in this application was not only that the coding overhead is 6 percent, but also that the existing satellite transponder equipment could be readjusted to accommodate the slightly higher bandwidth without modification. This allowed integration of the RS code into the TDMA environment in a very transparent fashion.

The Reed-Solomon code used is a RS(204,192) code on 8 bit symbols. For Gaussian noise conditions, this code will improve a raw bit-error rate of 10^{-4} to a corrected bit-error rate of 10^{-11}. The channel data rate is 51 megabits/second, with the user data rate being 48 megabits/second.

The development model of this system is constructed of standard ECL components. Encoding is performed in real-time using straightforward techniques. Decoding is performed by a dedicated, microcoded Galois field processor [24].

At these data rates, a decoder that operates in real-time on worst case data is not always the most cost-effective implementation. Instead, an approach known as "Best Case" or "Average-Case" Optimization can be used [19]. In this approach, the decoding engine can operate in real time under the expected channel conditions, but will not keep up with the incoming data if the channel becomes exceptionally noisy. A data-buffer (154 codewords in this case) is used to absorb the data-dependency of the decoder throughput, thus simulating real-time operation for the specified channel conditions.

The buffered decoder approach takes advantage of the fact that the complexity of decoding a received RS codeword is a function of the number of errors—the more errors, the more processing time is required. The performance of a buffered decoder is identical to that of a true real-time decoder except in those worst-case scenarios where buffer overflow occurs. In this event, some amount of uncorrected channel data will be passed to the user. The incidence of buffer overflow for a given system can be analyzed using queueing theory. As a practical illustration, the results obtained from the TDMA system codec are given in Fig. 8. The lower curve on this graph gives the theoretical relationship between raw BER and corrected BER for the RS code. The circles are data points measured during field testing of the codec unit. The difference between the theoretical and measured curves in the region between 10^{-2} and 10^{-3} raw BER is due to the occurrence of buffer overflow events. For raw BER less than about 7×10^{-4} the measured data agrees very closely with the theoretical performance.

The CCSDS Standard: Concatenated Block and Convolutional Code

This section reports on the Consultative Committee for Space Data Systems (CCSDS) Blue Book standard for Telemetry Channel Coding [7]. A schematic for the system is shown in Fig. 9. A space platform supplies a data source for which errors would prove damaging, for example, compressed video data. With such data, if any errors are not corrected, the de-compression algorithms will propagate the effect normally to the detriment of several successive frames. The data source is first encoded by a RS(255,223) encoder on 8 bit symbols. The output of the RS encoder is then interleaved on a symbol basis and input into the (2,1) K=7 convolutional encoder shown in Fig. 1.

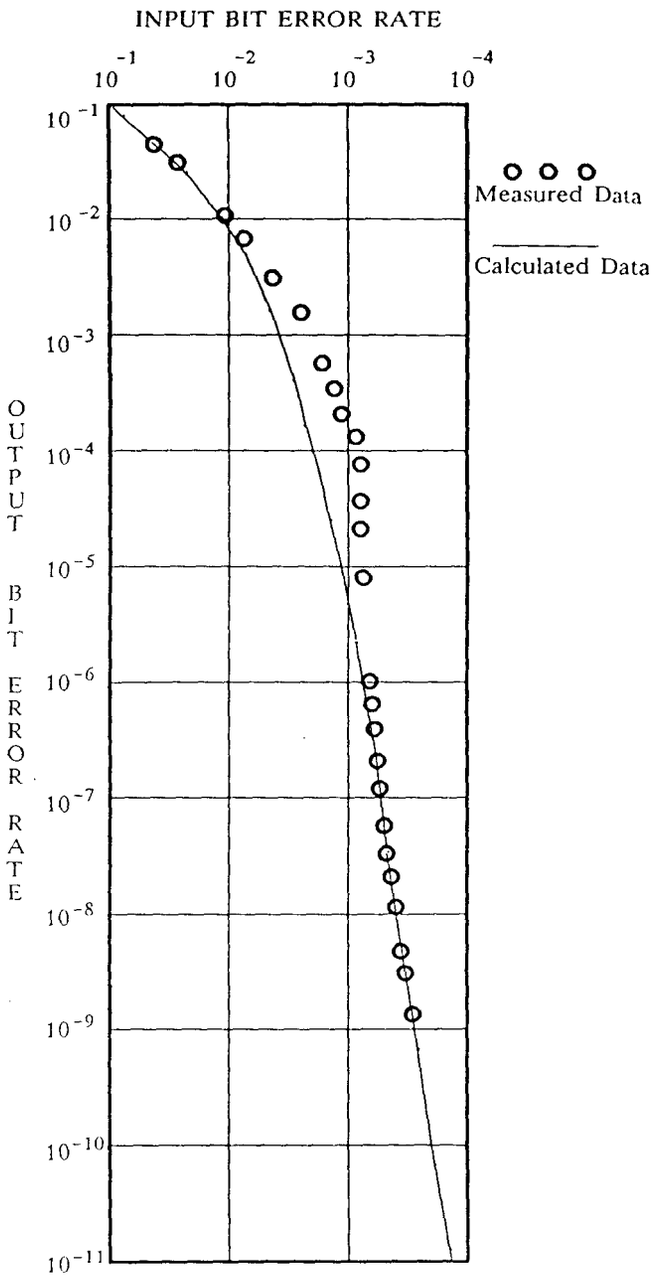

Fig. 8. Performance of the TDMA Codec.

The data rate is not defined in the standard. The RS code has been implemented to work at T1 (1.544 Mb/s) speeds and as high as 2 Mb/s.

The decoding reverses the above chain and is more complex. Figure 10 shows the theoretical performance of the convolutional code alone as opposed to uncoded. In fact, practical experiments have been performed for the scheme [21]. Figure 11, reproduced by kind permission of NASA JPL, shows some of the measured results.

The convolutional decoder uses the Viterbi Algorithm to process all the available soft decision data. This results in improved performance but, because of the small value of constraint length, the rate of descent is not that steep in either the practical or theoretical measurements. It is noticeable that in the measured case, the convolutional

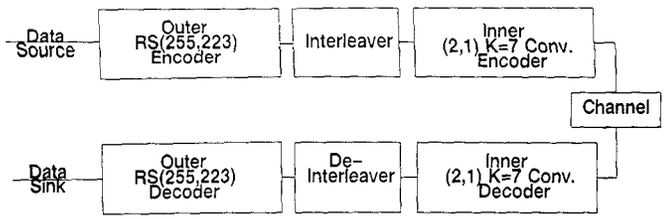

Fig. 9. CCSDS Concatenated Code Schematic.

code is offset from the theoretical by about 0.25 to 0.5 dB. This is suspected to be due to non-Gaussian effects on the channel.

Fortunately, the output from the convolutional decoder is well-suited to interleaved RS codes. The interleaving slices the bursts into shorter lengths that a codeword can handle with high probability. The symbol structure of the RS code effectively compresses the error bursts. For example, an error burst of 8 bits can affect 2 symbols at most. Finally, the RS decoder attempts to correct the errors.

Figures 10 and 11 show the theoretical and practical effect of the RS code. In the practical case, the curve is near vertical in the 3.25 to 3.4 dB region. This type of curve presents the system designer with a power budget, that is, if the designer provides a signal above a pre-set power, the data quality can be regarded as perfect.

In terms of complexity, several innovative techniques were used (and standardized) to reduce the RS encoder complexity [22,23]. The encoder, being on the space platform, is the critical component in terms of reducing hardware. It is interesting to note that these encoders require less hardware and less power than encoders for much "simpler" binary codes of the same redundancy but of lesser performance. The symbol nature of the RS codes and the algebraic nature of the encoding and decoding algorithms is well-matched to digital implementation.

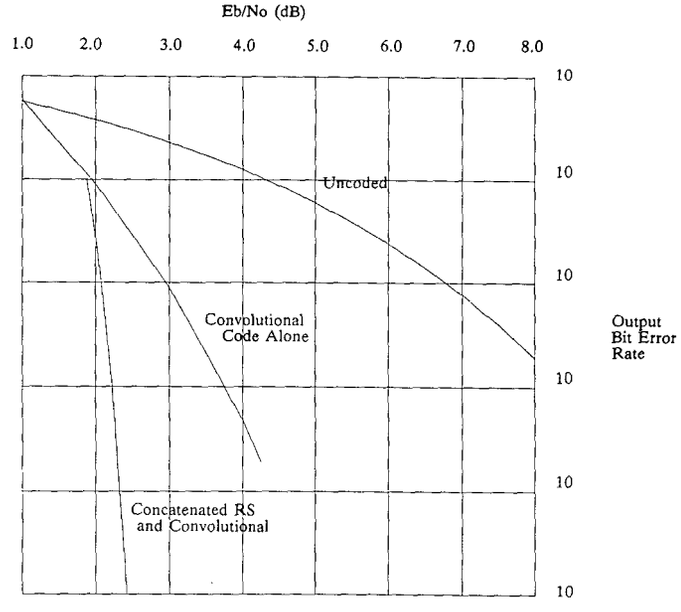

Fig. 10. Performance of RS(255,223) and (2,1), K=7 Conv. Code.

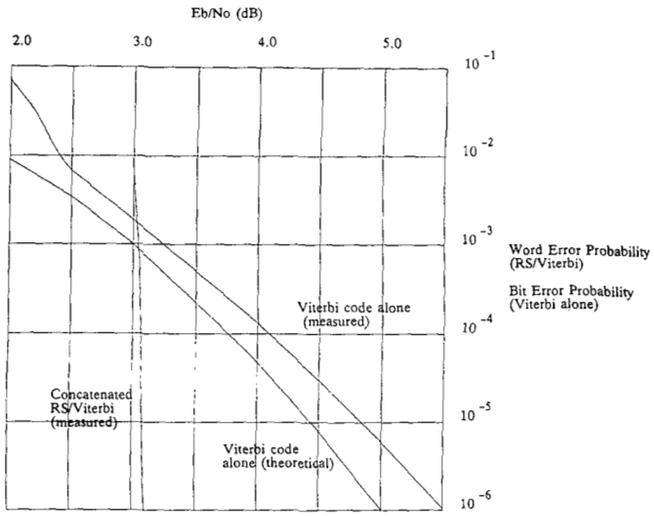

Fig. 11. Performance of the RS/Viterbi Concatenated Coding System. (By permission of the Jet Propulsion Laboratory [22])

The RS decoder consists of a sync acquisition subsystem and a Galois-field processor [24]. It is also interesting to note that CCSDS is now considering a Green Book recommendation for coding Space Station communications [25]. In this application, the links are a lot faster (300 Mb/s). At these speeds, convolutional decoding becomes difficult. The Green Book recommendation is to retain the RS code and drop the convolutional.

A Concatenated Reed-Solomon/ Binary Block Code System

In tactical military communications systems, a variety of error-control techniques are often combined into a single system.

One such system which has been recently developed [26] combines the use of a Reed-Solomon outer code, pseudo-random interleaving, and a binary inner code which is designed to impart certain spectral characteristics to the modulated signal.

The design of the binary inner code for this system is fairly involved. The system specifications impose a bias requirement on this code: the frequency of one's in the encoded data should be approximately one third of the total number of bits. Bias requirements such as this can arise when it is necessary for the receiver to recover the signal under very severe noise and interference. This restriction was satisfied by using the (24,12) extended Golay code, shortened to length (22,10), and then expurgated so as to include only 128 codewords, each of which has seven one's. It follows that the inner code has rate 7/22 and satisfies the bias requirement.

The overall inner coding technique can be categorized as a spread-spectrum method. Maximum likelihood estimation is used in the soft-decision inner code decoding algorithm, and an erasure indication is supplied to the outer Reed-Solomon decoder. The outer code is over GF(128) with 27 check symbols. The overall code rate of the concatenated codec is 0.25.

Pseudorandom interleaving is used to improve the

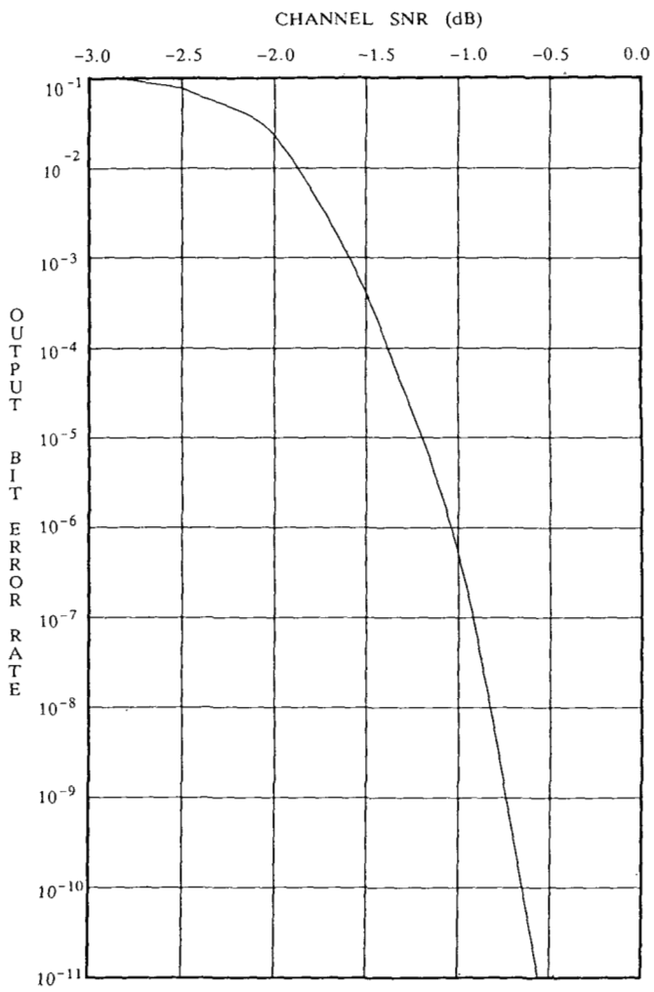

Fig. 12. Performance of the Model 7 Codec.

performance in the presence of interference. This form of interleaving is less suspectable to counter-measures than the more typical block or helical interleaving methods, for which effective periodic jamming signals can be designed.

The performance of this codec is shown in Fig. 12, which relates decoded BER to channel signal-to-noise ratio. The decoder provides usable output when the channel SNR is as bad as −1 dB. This sort of performance could also have been obtained using a low-rate, soft-decision convolutional code as the inner code; however, such a code would not satisfy the bias requirement. The use of the modified Golay code to meet this requirement represents part of the growing trend in which channel coding, modulation and synchronization are co-designed, rather than treated as separate system functions.

Conclusions

This article has introduced the changing strategy and tactics of error control. Certain classes of forward error correction codes and re-transmit protocols can fundamentally change system performance provided they are matched to the channel conditions and user require-

ments. In regard to the latter point, the article has discussed the properties and features of the most widely applicable techniques. The approach to this somewhat subtle area has been deliberately nonmathematical and some caveats must apply; there is a danger in relying too much on intuition and generalizations. Furthermore, the important area of forward error correction code implementation and algorithm design is only discussed in the widest of terms (although references are given). Some other important areas (for example, redundant signal sets) are only mentioned briefly; they are discussed elsewhere in this issue. Finally, ARQ protocols are only mentioned in relation to error control. Most commercially available ARQ protocols are embedded in sophisticated computer protocols of general utility.

It is hoped that the article will give the communications system designer a clearer view of the potential that coding can achieve and an appreciation of the changing trends in the application of error control.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Tech. Jour.*, vol. 38, pp. 61-656.

[2] J. L. Doob, correspondence, *Math. Rev.*, V. 10, p. 133, 1949.

[3] *New Directions in Communications Theory*, St. Petersburg, FL, April 1971.

[4] I. S. Reed, and G. Solomon, "Polynomial codes over certain finite fields," *Jour. Soc. Ind. Appl. Math.*, vol. 8, pp. 300-304, June 1960.

[5] G. D. Forney, "Concatenated codes," *Research Monograph no. 37*, M.I.T., 1966.

[6] E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York, 1968; Aegean Park Press, 1984.

[7] Consultative Committee for Space Data Systems, *Recommendations for Space Data System Standards: Telemetry Channel Coding "Blue Book,"* May 1984.

[8] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Info. Theory*, vol. IT-13, pp. 260-269, Apr. 1967.

[9] G. D. Forney, "Convolutional codes III: sequential decoding," *Tech. Report 7004-1*, Information Systems Laboratory, Stanford University.

[10] T. Le-Ngoc, K. Feher, and H. P. Van, "New modulation techniques for low-cost power and bandwidth efficient satellite earth stations," *IEEE Trans. Comm.*, vol. COM-30, no. 1, Jan. 1982.

[11] G. D. Forney, R. G. Gallager, G. R. Lang, F. M. Longstaff, and S. U. Qureshi, "Efficient modulation for band-limited channels," *IEEE Jou. of Selected Areas in Comm.* vol. SAC-2, no. 5, Sept. 1984.

[12] A. J. Viterbi, and J. Omura, *Principles of Digital Communications and Coding*, McGraw-Hill, 1979.

[13] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets—an overview," *IEEE Communications Magazine*, vol. 25, no. 2, Feb. 1987.

[14] I. M. Jacobs and E. R. Berlekamp, "A lower bound to the distribution of computation for sequential decoding," *IEEE Trans Info. Theory*, vol. IT-13, pp. 167-174, Apr. 1967.

[15] T. Kasami, T. Fujiwara and S. Lin, "A concatenated coding scheme for error control," *IEEE Trans. Comm.*, vol. COM-34, no, 5, May 1986.

[16] S. Lin and D. J. Costello, "A survey of various ARQ and hybrid ARQ schemes, and error detection using linear block codes," *IEEE Communications Magazine*, vol. 22, no. 12, Dec. 1984.

[17] D. Chase, "Code combining—a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets," *IEEE Trans. Comm.*, vol. COM-33, no. 5, May 1985.

[18] E. R. Berlekamp, J. Shifman and W. Toms, "An application of Reed-Solomon codes to a satellite TDMA system," *MILCOM 86*, Monterey, CA, Oct. 1986.

[19] E. R. Berlekamp and R. McEliece, "Average-case optimized buffered decoders," NATO Advanced Study Institute, Paris, July 11-22, 1983.

[20] L. R. Welch and E. R. Berlekamp, *Error-Correction for Algebraic Block Codes*, U.S. Patent Application No. 536951, Sept. 28, 1983.

[21] K. Y. Liu and J. J. Lee, "Recent results on the use of concatenated Reed-Solomon/Viterbi channel coding and data compression for space communications," *IEEE Trans. Comm.*, vol. COM-32, no. 5, May 1984.

[22] E. R. Berlekamp, "Bit-serial Reed-Solomon encoders," *IEEE Trans. Info. Theory*, vol. IT-28, no. 6, Nov. 1982.

[23] E. R. Berlekamp, *Bit-Serial Reed-Solomon Encoders*, U.S. Patent No. 4,410,989, Oct. 18, 1983.

[24] E. R. Berlekamp, *Galois Field Computer*, U.S. Patent No. 4,162,480, July 24, 1979.

[25] Consultative Committee for Space Data Systems, *Space Station: Application of CCSDS Recommendations for Space Data System Standards to the Space Station Information System (SSIS) Architecture "Green Book,"* Oct. 1985.

[26] E. R. Berlekamp, P. Tong, R. McEliece, R. J. Currie, and C. K. Rushforth, "An error-control code with an imbalance of ones and zeroes to provide a residual carrier component," *MILCOM 86*, Monterey, CA, Oct. 1986.

[27] CCITT Recommendation X.25, "Interface between data terminal equipment (DTE) and data circuit termination equipment (DCE) for terminals operating in the packet mode on public data networks."

[28] E. R. Berlekamp, "Hypersystolic computers," JASON Workshop on Advanced Computer Architectures, La Jolla, CA, July 1986.

[29] E. R. Berlekamp, (ed), *Key Papers in the Development of Coding Theory*, IEEE Press, 1973.

Elwyn R. Berlekamp received B.S., M.S., and Ph.D. degrees from the Massachusetts Institute of Technology in 1962, 1962, and 1964, respectively.

In 1973, Dr. Berlekamp founded Cyclotomics, Inc., an organization dedicated to the research, study, design, and implementation of high-performance encoding and decoding systems. Since 1982, he has been devoting most of his time to this enterprise.

Robert Peile received a B.Sc. degree in Mathematics with First Class Honors from the University of London 1976. In 1977 he received his MSc in Mathematics from Oxford University and in 1979 his D. Phil. in Mathematics from Oxford University.

At Cyclotomics, Dr. Peile is a member of the Senior Scientific Staff. He is involved in the development and configuration of specialized systems that utilize error correction and detection.

Stephen P. Pope received the B.S. degree with majors in Mathematics and Engineering from the California Institute of Technology, Pasadena, California, in 1978, and the Ph.D. degree in Electrical Engineering and Computer Science from the University of California, Berkeley, in February 1985.

After completing his studies at the University of California, Dr. Pope joined the staff of Cyclotomics, where he is the manager of VLSI design activities.

■