Formulation of Forward Error Correction Coding Recommendations for Future NASA Space Communications

Jon Hamkins, Leslie Deutsch, Dariush Divsalar, Sam Dolinar, Dennis Lee
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
{first.last}@jpl.nasa.gov

Frank Stocklin
Goddard Space Flight Center
Mailstop 450.0 Greenbelt, MD 20771
Tel: 301-286-6339
Fax: 301-286-1724
Frank.J.Stocklin@nasa.gov

John Wesdock, Chitra Patel
ITT Industries
Advanced Eng. & Sci. Division
12975 Worldgate Drive
Herndon, VA 20170
Tel: (703) 668-6332
{first.last}@itt.com

Abstract—NASA has undertaken a study to recommend and justify Coding, Modulation, and Link Protocol (CMLP) designs for the Space Communications and Networking (SCaN) office (see companion paper [1]). This paper reports on the coding part of the CMLP study, which is chartered with identifying the forward error correction (FEC) codes suitable for NASA space exploration and science missions through 2030.

TABLE OF CONTENTS

1. INTRODUCTION

The purpose of forward error correction (FEC) coding, or channel coding, is to reduce the power needed in order to achieve a given error rate on a communications channel. The CMLP study included a coding subteam to analyze and select channel codes appropriate for the short, medium, and long term needs of NASA.

We began the study by compiling a comprehensive catalog of over 150 candidate FEC codes. Any code with potential application to a link in NASA's Space Communication and Navigation Architecure recommendations for 2005-2030 [2] was included. For each code, we recorded the code length, code rate, decoding latency, codeword error rate and bit error rate performance, standardization status, flight heritage, maturity, and encoder/decoder complexity. Coding standards such those of the Consultative Committee for Space Data Systems (CCSDS), Digital Video Broadast Satellite 2 (DVB-S2), and Institute of Electrical and Electronics Engineers (IEEE) 802.11 and 802.16 were included in the catalog. Additionally, an attempt was made to include any other code that reasonably has application in one or more of the reference links. The code catalog includes uncoded, convolutional, Reed-Solomon (RS), concatenated RS/convolutional, turbo, serially concatenated convolutional, Bose-Chaudhuri-Hocquenghem (BCH), low-density parity-check (LDPC), turbo product, concatenated BCH/LDPC, and cyclic redundancy check (CRC) codes.

The code selection approach involved the following steps, for each reference link under consideration: (1) Compute the bandwidth used by the recommended modulation at the desired data rate, using uncoded transmission, (2) Compute the minimum code rate r available, using step 1 and the total bandwidth available on the link, (3) Compute the maximum input block size k, using the latency requirement, (4) Sort the code catalog based on the rate r, (5) Sort the code catalog by input block size k, among codes with eligible rates, (6) Identify top performing codes within the (k, r) constraints. Finally, the top selections were further narrowed to a small set of recommended codes that meet the requirements of all links, based on an analysis of the remaining figures-of-merit.

The coding study concluded by selecting various uncoded, convolutional, turbo, and LDPC codes for a set of reference links. Turbo codes of rates 1/6, 1/4, and 1/3 are recommended for low code rate applications, accumulate-repeat-jagged-accumulate (AR4JA) LDPC codes of rate 1/2, 2/3, and 4/5 are recommended for higher rate applications, and the C_2 LDPC code of rate 7/8 is recommended for bandwidth constrained links. The constraint length 7, rate 1/2 convolutional code is recommended for low data-rate real-time links and complexity-limited applications, while uncoded transmission is recommended wherever the link signal-to-noise ratio (SNR) supports it. Legacy applications of convolutional and concatenated Reed-Solomon/convolutional are also rec-

1-4244-1488-1/08/$25.00 ©2008 IEEE
IEEEAC paper # 1453, Version 2, Updated December 14, 2007

ommended.

As it turned out, the selection approach above resulted in codes that are all existing standards or experimental specifications of the Consultative Committee for Space Data System (CCSDS).

2. CODES CONSIDERED BY THE STUDY

Table 5 in Appendix A lists all candidate coding schemes considered in the CMLP study. Later sections describe the main classes of coding techniques.

The CMLP forward error correction coding catalog contains 167 specific codes from nine code groups. Represented code groups are of two broad types: classic (legacy) codes, including convolutional codes (CC), Reed-Solomon (RS) codes, RS+CC concatenated codes, Bose-Chaudhuri-Hocquengham (BCH) codes, and cyclic redundancy check (CRC) codes; and modern iteratively decoded codes, including parallel concatenated convolutional codes (turbo codes), serially concatenated convolutional codes (SCCC), turbo product codes (TPC), and low-density parity-check (LDPC) codes. A general description is provided here for each of these code groups as well as pros, cons and typical applications.

Classic Codes

Convolutional Codes—Convolutional codes are codes which perform a convolution of the input data stream with the encoder's impulse responses. Standard texts (e.g., [3]) describe this and the other classic codes discussed here. Convolutional codes map k bits into n symbols based not only upon the current k information bits but also all previous information bits (or as practical). Convolutional codes can be recursive or non-recursive and systematic or non-systematic.

A convolutional code's effectiveness is fundamentally limited by the constraint length K of its convolution. Convolutional codes with arbitrarily long constraint lengths can approach the Shannon limit with maximum-likelihood (Viterbi) decoding. Practical convolutional codes are limited to reasonably small constraint lengths (e.g., $K = 7$), because the decoding complexity increases exponentially with K.

Convolutional codes have been used extensively by NASA and by all of the communications industry. This successful legacy of use and the equipment and infrastructure base it has created is probably the single most important attribute in favor of convolutional codes. Other favorable attributes include: 1) Code synchronization is simple and quick as compared to most block codes, and it can be performed automatically by the decoder without the need to devote additional overhead to a synchronization marker. 2.) These codes have very low latency, on the order of one constraint length for encoding and a handful or two of constraint lengths for decoding. 3) Error rates with Viterbi decoding diminish exponentially with increasing signal-to-noise ratio, and these codes show no signs of an error floor.

Attributes of convolutional codes that are not favorable include: 1.) Convolutional codes dramatically underperform modern high-performance iteratively decoded codes such as LDPC codes and turbo codes; 2.) Convolutional decoders are not naturally implemented with a parallel architecture, a factor that limits their speed particularly when these codes are used as constituents of iteratively decoded codes (turbo codes and SCCCs).

Convolutional codes are common in all areas of communications including space communications and terrestrial mobile communications. Convolutional codes have been used extensively and very successfully on many NASA missions including the Hubble Space Telescope and Voyager. Two more recent NASA deep space missions, Mars Pathfinder and Cassini, opted for an exceedingly complex-to-decode convolutional code with constraint length $K = 15$, in order to gain increased performance at relatively low data rates from deep space.

As better performing modern iteratively decoded codes continue to emerge and become commonplace, fewer communication industry areas are choosing convolutional codes over the newer codes. While NASA will continue to support existing convolutional codes for some time, it is expected that there will be some future transition to more efficient codes.

Reed-Solomon (RS) Codes—Reed-Solomon (RS) codes are nonbinary systematic codes which introduce (NK) parity symbols for every K symbols of information. Each RS symbol is formed from multiple bits. RS codes can detect up to (NK) error symbols, or can correct up to $(NKe)/2$ error symbols in combination with e erased symbols, for any $0 < e < NK$.

Reed-Solomon codes are maximum-distance-separable (MDS), and therefore are capable of correcting the maximum possible number of errors (or combination of errors and erasures) among all codes of given K and N. Furthermore, RS encoders and traditional RS decoders[1] are relatively low-complexity and can be implemented in hardware at very high data rates. While this seems to be an ideal combination of code attributes for any application, it is well known that RS codes perform very poorly on many useful channels such as the additive white Gaussian noise (AWGN) channel. There are two primary reasons for this disappointing performance. First, a traditional RS decoder bases its decisions only on hard-limited output from the channel, ignoring useful reliability information (worth about 2 decibels (dB) in AWGN). Second, there is a channel error magnification effect because

[1] Recently, enhanced RS decoding algorithms have been developed to utilize soft channel information, but these are high-complexity, low-maturity algorithms that would disqualify RS codes from being considered as "legacy codes." In fact, these algorithms are of sufficiently low maturity (compared to other newer codes and decoders) that RS codes with enhanced soft decoding were not evaluated for this study in the "modern codes" category either.

each isolated channel symbol error (e.g., a bit error with binary phase-shift keying (BPSK) modulation) corrupts a larger nonbinary RS symbol. Together these two effects account for several dB of performance loss when RS codes are applied to soft-output channels such as AWGN.

RS codes can be efficient when applied to channels that are inherently bursty and produce hard-limited output. An RS code can also be useful for a soft-output channel afflicted with white noise if it is concatenated (as an outer code) with an inner code (such as a convolutional code) that is more suitable for exploiting the characteristics of the channel corruptions (see next subsection).

RS codes with a symbol size of 8 bits and $(N, K) = (255, 223)$ and (255, 239), as well as shortened versions of these codes, are currently supported by GN, SN and DSN.

RS+CC Concatenated Codes—A concatenation of a Reed-Solomon outer code with a convolutional inner code (RS+CC concatenated code) is a classic code that exploits the MDS properties and the large block size advantages of the RS outer code, together with the ability of the convolutional inner decoder to efficiently extract soft information from the channel with low complexity. The performance of RS+CC concatenated codes is characterized by a steeply falling error rate curve. The slope of this curve is ideally determined by the block size of the RS outer code, which is nearly two thousand information bits for classic RS codes using 8-bit symbols, and for useful shortened versions of these codes. The burstiness of errors output from the inner Viterbi decoder greatly reduces the error magnification effect that ordinarily would diminish the effectiveness of an RS code with 8-bit symbols if it were connected directly to a white noise channel. However, because the Viterbi decoder error bursts are unpredictable in length, and occasionally longer than several RS symbols, classic RS+CC concatenations generally include a block interleaver between the inner and outer code in order to break up long Viterbi decoder bursts into smaller pieces distributed among multiple RS codewords. This interleaving improves the RS+CC concatenated code's performance, but at the expense of increasing its block size and therefore its latency. The overall information block size is nearly nine thousand bits for the classic RS+CC concatenated code built from an 8-bit RS outer code, a $K = 7$ convolutional inner code, and a depth-5 interleaver.

The deteriorated performance of an RS+CC convolutional code without interleaving, as compared to the same code with sufficient interleaving to break up the Viterbi decoder error bursts, is similar in principle to the error floor phenomenon that plagues turbo codes at very low error rates. The longer the constraint length (and hence longer error bursts) of the convolutional code, and the smaller the size of the RS code, the more the performance of the RS+CC concatenated code will be determined by that of its inner convolutional code and not by the complementary strengths of the overall concatenation. However, this degradation of performance shows up as a fairly uniform diminishment of slope of the concatenated code's error rate curve, unlike the sharp transition in slope in the performance curve of a turbo code at the beginning of its error floor region. Furthermore, because the typical error bursts for a $K = 7$ convolutional code are still small compared to the size of an 8-bit RS code, the performance curve of this particular RS+CC concatenation without interleaving is still much steeper than that of a turbo code past the start of its error floor.

Even with ideal (infinitely long) interleaving, RS+CC concatenated codes are not capable of approaching the Shannon capacity limit of performance more closely than about 2 dB, unless the inner convolutional code's constraint length is impractically long or the combined decoding of the inner and outer code is impractically complex. A highly complex RS+CC concatenated code was designed to support NASA's Galileo mission to Jupiter after Galileo was forced to transmit exclusively through its low-gain antenna at extremely low data rates (around 100 bps). This code featured a very long constraint-length convolutional inner code $(K = 14)$, a variable-rate RS outer code, a long interleaver (depth-8), and four stages of alternating decoding between the inner and outer codes (a form of iterative decoding passing hard rather than soft information during the iterations). With these enhancements (generally impractical except at Galileo's low data rates), the gap to capacity was shaved to about 1 dB. But this performance still falls about 1/2 dB short of that of modern turbo or LDPC codes of similar sizes and rates and much lower complexity.

As with all long block codes, RS+CC concatenated codes require accurate synchronization to determine the starting position of a codeword from among thousands of possible locations. As compared to similar synchronization requirements for other long block codes (including turbo codes, TPCs, SCCCs, and LDPC codes), synchronization for RS+CC concatenated codes can be somewhat less burdensome due to the inner convolutional decoder's self-synchronizing capability and the decoupling of the inner and outer decoders. This allows codeword synchronization for RS+CC concatenated codes to be accomplished using Viterbi decoded bits rather than a larger number of lower-reliability raw channel symbols. In this case, synchronization performance is not necessarily improved, but the required processing is simplified. RS+CC concatenated codes are currently widely used for GN, SN and DSN.

BCH Codes—Bose-Chaudhuri-Hocquengham (BCH) codes are classic binary codes that can be designed to correct small to moderate numbers of bit errors without excessive encoding or decoding complexity. As with many other classic codes, reasonable-complexity decoding algorithms for BCH codes use hard inputs. For this reason, BCH codes are generally a poor choice to apply directly to a soft-output channel such as a Gaussian noise channel. Additionally, BCH codes have

generally taken a back seat to RS codes as an outer code in a concatenated system, due to an RS code's capability to correct more bit errors when both types of codes are constrained to similar complexities.

In modern times, BCH codes are used in the DVB-S2 standard as an outer code to an inner LDPC code. The purpose of the BCH code in this setting is to lower an otherwise unacceptably high error floor due to frequent errors of very low weight produced by the LDPC decoder. For the purposes of the CMLP study, BCH codes were evaluated only in the context of this specialized application, in conjunction with the particular LDPC codes designed for the DVB-S2 standard. More recent designs of LDPC codes, such as C_2 and the AR4JA family, are not susceptible to the low-weight error events that would be correctable by a BCH code. In this case, concatenation of the LDPC code with a BCH code would be superfluous and would only serve to reduce the power efficiency of the overall code without any appreciable lowering of its error rate.

CRC Codes— Cyclic redundancy check (CRC) codes are codes that append a fixed number of parity bits to large information blocks of varying lengths for the purposes of error detection only. Typical CRC codes use 16-bit, 32-bit, or somewhat longer parity sequences. A CRC code is usually used as an outer code concatenated with an inner error-correcting code (which may itself be a concatenation of two or more constituent codes).

To first order, the probability that an erroneous codeword escapes detection by an outer CRC code is roughly the same as the probability that its parity bits agree with an equal number of random bits. The conditional undetected error probability of an m-bit CRC code is roughly 2^m if the inner codeword's typical error patterns are long and varied. Good CRC codes are generally designed to offer guaranteed detection of small numbers (e.g., up to 3) of bit errors, on the assumption that higher-weight error patterns occur with much lower probability. This design feature greatly enhances a CRC code's detection performance with uncoded data, but it is nearly worthless when used with a powerful inner error-correcting code that is likely to make either no bit errors or many bit errors in bunches.

Because CRC codes are not used to correct any errors, they cannot improve the ability of the overall coding system to approach the Shannon limit of performance. In fact, the power efficiency of the overall code is reduced by the rate of the CRC code, with no improvement in error-correcting capability. When the CRC code is attached to large inner codewords or data frames thousands of bits long, the CRC code's overhead penalty is very small and is tolerated in return for its ability to reliably detect errors. However, the performance penalty for using a CRC code becomes non-negligible if it is used to protect smaller codewords or frames on the order of a few hundred bits or less.

Encoding and decoding of CRC codes is accomplished using simple linear feedback shift registers, and this encoding and decoding architecture does not change with the size of the frame being protected. This invariance of coding and decoding architecture to code block size is a primary reason why CRC codes are generally preferred over other codes with equal or better error detection capabilities. Often, however, the inner error-correcting code has error detection capabilities of its own, and in such cases the CRC code becomes a useless appendage that reduces power efficiency without offering improved error detection. A CRC code is not needed to improve the inherent error detection capabilities of the classic (255, 223) RS code (or the corresponding RS+CC concatenated code), and CRC codes up to at least 32-bits do not appear to improve the native error detection capabilities of well-designed LDPC codes, such as C_2 or the AR4JA family.

Modern Codes

Turbo Product Codes (TPC)— A product code is obtained from constituent (N_1, K_1) and (N_2, K_2) codes by filling an N_1 by N_2 rectangular array of coded bits with: 1) a K_1 by K_2 rectangular array of information bits; 2) a K_1 by (N_2K_2) rectangular array of parity bits computed by applying the (N_2, K_2) code's encoding rule to each of the K_1 rows of information bits; and 3) an (N_1K_1) by N_2 rectangular array of parity bits computed by applying the (N_1, K_1) code's encoding rule to each of the K_2 columns of information bits and (N_2K_2) columns of parity bits computed in the previous step. This product code maps K_1K_2 information bits into a total of N_1N_2 coded bits. An identical product code results if the column encoding is done first and the row encoding second. If one of the constituent codes is itself a product code, the resulting code is a product code in three or more dimensions.

Product codes are classic codes that have not found many useful applications until recently, due to poor minimum distance for relatively large block size and the unavailability of soft maximum-likelihood decoding algorithms. However, the iterative decoding revolution launched by turbo codes also sparked a revival of product codes. Product codes can be decoded iteratively by alternating the decoding of rows and columns, and passing soft extrinsic information between the row and column decodings in the manner of turbo decoding. A classic product code decoded in this manner is called a turbo product code (TPC), and is now regarded as a modern code.

Encoders and decoders for turbo product codes can be implemented at high speed, because the individual row and column encodings and decodings can be performed in parallel. Typical constituent codes of a TPC are single-parity-check codes, Hamming codes, and extended Hamming codes, with small minimum distances of 2, 3 and 4, respectively. Such constituents are selected because they are fairly easy to decode individually, and their relatively high rates keep the rate of the product code (equal to the product of its con-

stituents' rates) from being unreasonably low. However, since the minimum distance of the product code is the product of its constituents' minimum distances, two-dimensional product codes built from such constituents will include many incorrect codewords within distance 16 of the true codeword, and the decoder's error rate will reach a low-slope error floor region where further improvements are limited by the difficulty of distinguishing among these relatively close neighbors. TPCs in three dimensions built from extended Hamming constituent codes can achieve a minimum distance of 64, and their error rate curves fall off much more steeply even when the minimum-distance neighbors dominate the performance. As a result, their error floors may be imperceptible in many applications. However, TPCs of three (or more) dimensions are more complex, their iterations require an extra round (or more) of decoding, their overall rates are fairly low, and their overall block sizes are quite large.

TPCs share many of the same features as other modern codes including turbo codes, SCCCs, and LDPC codes. As long block codes, they have long latency relative to classic convolutional codes, and they require accurate synchronization among thousands of possibilities to determine the starting location of a codeword. As long codes that can achieve near-maximum-likelihood performance via iterative decoding, some TPCs can approach the Shannon performance limit as closely as turbo and LDPC codes, particularly for very large block sizes on the order of tens of thousands of bits. Generally speaking, turbo, SCCC and LDPC codes offer much more flexibility for designing near-optimal codes at a wide range of rates and sizes down to a thousand bits or lower, while only a few scattered point designs of TPCs are equally near-optimal unless the size of the code is extremely large.

Turbo Codes— Turbo codes[4] are parallel concatenations of two or more simple recursive convolutional codes, used to encode differently permuted versions of the same information sequence. The different permutations of the input information bits are accomplished by one or more interleavers. Turbo codes are decoded iteratively by passing soft extrinsic information between two relatively simple convolutional decoders tasked to decode the constituent codes separately. If the information block is reasonably large and the interleaver(s) sufficiently random, the iterative turbo decoder achieves nearly the same performance as an impossibly complex maximum-likelihood decoder for the same code. Furthermore, turbo codes can approach the Shannon capacity limit of performance with well-designed constituent codes and interleaver(s).

Unlike turbo product codes, which are a classic code structure to which iterative decoding principles are applied, turbo codes (as well as serially concatenated convolutional codes and LDPC codes) are modern codes that were designed from the start to be decoded iteratively. For example, the recursive property of the turbo code's constituent convolutional codes is critically important for its near-optimal performance, but this property has only a minor impact on the performance of these same convolutional codes decoded classically.

Practical turbo codes are generally limited to two constituent codes, because of the need for multiple interleavers, the increased length of an iteration cycle, and the lower rate of the overall turbo code, when more than two constituents are used. However, parallel concatenations of two constituent codes are susceptible to an error floor, where the near-optimal performance of the overall turbo code breaks down and further reductions in error rate are limited by the properties of the weak constituent codes. With good choices of constituent codes and interleaver, this error floor can be driven low enough for many applications, e.g., CWER in the range of 10^6 to 10^8 and BER an order of magnitude lower, but not sufficiently low for applications that require error rates a few orders of magnitude lower than this.

Even when limited to two constituents, the natural rates of parallel concatenations without puncturing are 1/3 and lower. Higher turbo code rates can be produced by puncturing some of the constituent decoders' outputs, but excessive puncturing can be detrimental to performance. Most useful turbo codes have been developed for code rates 1/2 and lower.

Good constituent codes for turbo codes have very short constraint lengths (e.g., $K = 3$ to 5), which makes them even simpler to decode than a classic medium-constraint-length convolutional code with $K = 7$ (even allowing for the turbo decoder's requirement that its constituent decoders produce soft rather than hard outputs). However, the turbo decoder's overall complexity is much higher than that of the $K = 7$ convolutional code, due to its needs for two such constituent decoders, for performing multiple iterations, for processing a large block of data at once, and for interleaving and deinterleaving the soft outputs from each constituent decoder during the course of each iteration. On the other hand, the complexity of turbo decoding is significantly lower than that of Viterbi decoding of the long-constraint-length ($K = 15$) classic convolutional code used by Mars Pathfinder and Cassini.

Turbo codes need to encode reasonably large blocks of information (e.g., a thousand or more bits) in order to achieve near-optimal performance commensurate with their block sizes. This contributes to high latency, and a need for accurate code block synchronization as discussed previously for TPCs.

Turbo codes are currently in use on NASA's Mars Reconnaissance Orbiter (MRO) and are supported by the DSN.

Serially Concatenated Convolutional Codes (SCCC)—A serially concatenated convolutional code (SCCC) [5] is a serial concatenation of two codes similar in concept to the classic RS+CC concatenation. The inner and outer codes of an SCCC are both short-constraint-length convolutional codes, typically $K = 3$ for the outer code and $K = 3$ to 5 for the inner code. The SCCC's inner convolutional code is recur-

sive (as are the parallel constituents of turbo codes). Between the inner and outer codes is an interleaver that resembles the random-like interleaver of turbo codes rather than the regular rectangular interleaver of RS+CC concatenated codes.

As with turbo codes, an SCCC is a modern code structure that was never considered useful or practical until iterative decoding algorithms were developed to decode it effectively and with reasonable complexity. A turbo code can be regarded as a special type of SCCC with a simple repetition code replacing the SCCC's outer convolutional code and its interleaver obeying some additional constraints.

The outer code of an SCCC generally has minimum distance at least 3, and this property eliminates the appearance of error floors at error rate levels typical of the error floors of turbo codes constructed from only two constituents. Furthermore, the outer convolutional code achieves this higher minimum distance without lowering the overall code rate as much as a turbo code's rate is lowered when it has more than two parallel constituents. Also, the serial combination of two nontrivial codes offers greater flexibility for puncturing either or both constituents to achieve higher rates while not sabotaging the near-optimality of the code's performance.

On the other side of the ledger, it is difficult to design SCCCs with decoding thresholds as low as those of turbo codes, and their decoding complexity is somewhat higher.

Low-Density Parity-Check (LDPC) Codes—LDPC codes [6] are old codes but not classic. Having been invented by Gallager nearly a half-century ago, they lay dormant for many decades until similarities were noted between Gallager's code constructions and iterative decoding methods, and those of Berrou et al. in their more recent invention of turbo codes. Modern LDPC codes have been re-engineered and optimized in many directions over the past decade since their rediscovery.

An LDPC code is defined by a sparse parity-check matrix containing only a few 1s in each row and column. This parity-check matrix can be represented by a sparsely connected graph introduced by Tanner. The Tanner graph also describes the paths along which messages are passed when the LDPC code is decoded iteratively. There are individual nodes in the Tanner graph for all of the coded bits and code constraints, and this feature enables extreme parallelization of the decoder's operations within each iteration. This contrasts with the time-sequential forward and backward message passing that takes place within each iteration on the trellis graph that represents the constituents of a turbo code or SCCC. This massive inherent parallelizability is a major advantage of the LDPC decoding algorithm, allowing LDPC decoding speeds to be limited mainly by the amount of hardware that can be practically assembled to perform primitive message passing operations in parallel.

In addition to providing the best potential for achieving high decoding speeds among iteratively decoded codes, LDPC codes also offer more degrees of design freedom compared to turbo codes and SCCCs and especially TPCs. This has enabled LDPC code designers to trade off decoding threshold, error floor performance and other attributes more effectively than for these other modern codes. Specific LDPC codes have been designed to approach microscopically close to the Shannon limit of performance, and there is no theoretical limitation on how low their error floors can be pushed. A decade of improving LDPC code design methods has resulted in codes of a wide range of practical sizes and rates that perform reasonably close to the Shannon limit down to error floor levels that are virtually undetectable.

Early LDPC code designs were highly unstructured, because random-like connections in the Tanner graph provide a statistically sound method to generate ensembles of good LDPC codes. However, unstructured designs lead to impractical decoding, due to the difficulty of properly routing messages in a large randomly connected (albeit sparsely connected) graph, notwithstanding the fact that the number of computations needed to generate each message does not increase if the connections are unstructured. More recently, quasicyclic LDPC codes have been designed from a small template graph (protograph) and a selection of circulant permutations. The Tanner graphs of quasicyclic LDPC codes have more regularly structured connections that simplify the LDPC decoder's architecture.

Despite their many intrinsic advantages, LDPC codes have not totally displaced turbo codes. The realm of rates less than 1/2 where turbo codes work best is also where good LDPC code designs become more difficult. At low rates, an LDPC code's Tanner graph acquires more connections as additional parity-check constraints are added. Furthermore, iterations take a very long time to converge, since each input symbol from the channel has very low reliability due to its highly diluted signal-to-noise ratio. In contrast, iterations proceed more quickly for low-rate turbo codes or SCCCs, since their constituent convolutional decoders use aggregated messages from several weak channel symbols to label the branches of their decoding trellises before starting their iterations.

Finally, turbo codes and SCCCs have an advantage over LDPC codes in their inherent ease of encoding. An LDPC code is defined via its sparse parity-check matrix, but the corresponding generator matrix for encoding the code is not sparse. This is in contrast to turbo codes and SCCCs, with their constituent short-constraint-length (low-density) convolutional encoders. However, recent structured LDPC code designs, especially quasicyclic LDPC codes, have made high-speed encoders feasible for LDPC codes as well.

3. INITIAL CODE SELECTIONS

The purpose of coding is to reduce the power needed in order to achieve a given error rate. As such, power efficiency is

generally the dominant Figure of Merit (FOM) in the comparison of the various codes. However, any of the other FOMs (spectral efficiency, latency, user burden, etc.) could prevent the use of the most power efficient codes. For example, a link with strict latency requirements can disallow the use of any code with a very large blocklength, because the time to receive and decode a long block exceeds the allowable latency. Or, the available bandwidth of a link may restrict the code rates that may be used, because for a constant data rate lower rate codes use inversely proportionally more spectrum.

The initial code selection procedure is performed after the final modulation selection process has completed. In particular, for each link under consideration, we assume the use of the CMLP-recommended modulation (see companion article [7]). The spectral efficiency of this modulation allows us, then, to compute the eligible code rates, as we describe below.

Constraints Relevent to Code Selection

The bandwidth assignments (allocations) for various near Earth and deep space bands are given in Table 1.

Table 1. Bandwidth Assignments.

Band	Application	Bandwidth* Assignment (allocation)
S-band	Forward or return	6 MHz
S-band	Launch	10 MHz
X-Band	Near Earth forward	150 Mhz
X-band	Deep space, non-efficient modulations	4 MHz
X-band	Deep space, with efficient modulation	50 MHz
Ka-band	Near Earth return	650 MHz
Ka-band	Deep Space	500 MHz

*The bandwidth is measured by SFCG conventions as the 99% bandwidth metric for near Earth, and the 25 dB down metric for deep space.

The latency requirements are given in Table 2.

Table 2. Latency requirements.

Link Type	Application	Decoder Latency Requirement
Voice	Near Earth	100 ms
Voice	Lunar	250 ms
Non-voice	Any	N/A

Selection Procedure

These blocklength and bandwidth restrictions suggest an initial selection process that eliminates codes from consideration that would violate those restrictions:

1. *Compute bandwidth used by recommended modulation at specified data rate, uncoded.*
For a link using a given data rate R b/s and modulation with spectral efficiency η b/s/Hz, uncoded transmission uses a bandwidth of $B = R/eta$ Hz.
2. *Compute minimum code rate available, using step 1, and total bandwidth available.*
The transmission will meet a given bandwidth assignment (allocation) B_a Hz only if the code rate satisfies $r \geq B/B_a$. The values of B_a used by the study are given in Table 1.
3. *Compute maximum input block size k, using latency requirement.*
The decoding latency is the difference between the time a bit is decoded and the time its encoded version first begins arriving at the receiver. For a block code, this includes the time for a whole codeblock to arrive at the receiver plus the time it takes to decode it. For a convolutional code, it is the time for a number of bits to arrive that is equal to the traceback depth of the Viterbi decoder plus the time to perform a traceback operation. Given the high-speed decoders that exist today, the study assumed that the latency is dominated by the time it takes to receive the relevant bits to decode.
In order for a block code to meet a latency constraint on a link using a given data rate R b/s and having a latency requirement T_l s, the input block size k of the block code must satisfy $k \leq T_l \times R$. The values of T_l are given in Table 2.
4. *Sort code catalog shown in Table 5 based on rate r. Eliminate those with disallowable values of r.*
5. *Sort code catalog by input block size k, among codes with eligible rates. Eliminate those with disallowable values of k.*
6. *Identify top performing code(s) within (k, r) constraints based on performance, complexity, and maturity.*
7. *Narrow selections using, FOM analysis, to a small set of candidate codes that work for all links.*

This procedure may be carried out on each of the links identified in the SCaN architecture [1], as clustered together in [2]. Although these links are already quite numerous, we found it necessary to further partition the links, by data rate, in order to assure the capture of all latency and bandwidth constraints. For example, in [2], one link listed is an operational forward S-band link with a data rate of "$\leq$ 60 kbps." If it were exactly 60 kbps, then the latency requirement in step 3 above would require that the blocklength satisfy $k \leq T_l \times R = 6000$. However, an 18 kbps link would also fall into the "$\leq$ 60 kbps" category, but in that case the blocklength constraint is the more stringent $k \leq T_l \times R = 1800$. The two cases are sufficiently different that different coding solutions would be recommended. To ensure that different data-rate-dependent link drivers were captured, we partitioned the S-band data rates into ranges: 18 – 100 kbps, 100 – 300 kbps, 300 – 4800 kbps, and 4.8 – 6 Mbps.

Initial Code Selections

Following the first six steps above resulted in the first-stage select of codes shown in Appendix B, in Table 6, Table 7, Table 8, and Table 9. These are the codes among those that meet the bandwidth and latency requirements that have the highest power efficiency and acceptable complexity and maturity.

As can be seen, a relatively small set of code candidates covered all links:

- Legacy codes: uncoded, (7,1/2) convolutional, and BCH (63,56)
- AR4JA & C_2 — nearly the entire family of CCSDS orange book codes: (1024,1/2), (4096, 1/2, 2/3, 4/5), (16384, 1/2, 2/3, 4/5), C_2
- Turbo — CCSDS blue book codes of longest and shortest lengths: (1784, 1/4, 1/6), (8920, 1/2, 1/3, 1/4)
- Turbo Product codes — 2D and 3D versions: TPC(128,120)^2, TPC(16,11)^3, TPC(H64$\times$H32$\times$S32)
- F-LDPC: (16k, 2/3), (16k, 8/9)

4. FIGURES OF MERIT ANALYSIS

For each link, we ranked the codes surviving the first stage selection process by each of the ten FOMs. For each link, the FOMs are weighted to arrive at a final FOM score. A couple of observations are in order regarding these weights. First, because the initial selection procedure weeded out those codes that did not meet the bandwidth constraint, the remaining codes all meet the bandwidth constraint, and so there may be limited value in preferring one code over another with respect to this FOM. This is true even when the bandwidth constraint is very important or stringent. For example, if the bandwidth constraint forces the code rate to be 7/8 or higher, then the vast majority of codes from the catalog are eliminated from consideration in the initial selection process, but those that are remaining are not preferred over one another on the bases of spectral efficiency because all remaining codes meet the constraint. Therefore, the weighting of spectral efficiency (and latency) are quite low in the final FOM ranking and analysis.

A second FOM consideration is that the links do not use the same FOM weightings. This is a consequence of different mission screnarios giving rise to different priorities. For example, an uplink for an outer planets mission would weight the power efficiency FOM higher than a LEO mission, because the outer planet mission may have a much harder struggle to meet its data rate requirements without building significantly enhanced ground infrastructure, compared to a LEO mission.

We now describe how the ranking was done for each of the ten FOMs.

1. Supports legacy missions
The highest rank was assigned to codes that are currently flying on missions. The next highest rank was given to codes that plan to use the code, and the lowest rank was given to all other codes.
2. Spectral utilization
The ranking of spectral utilization was based on the entry in the table corresponding to "bandwidth used," which itself is a function of the given data rate and modulation spectral efficiency, and the rate of the code under consideration. Therefore, the ranking of the codes is a ranking of the code rates from highest to lowest. Since the first stage select process has already yielded a set of candidate codes of approximately the same code rate, there is no need to give additional weighting to spectral utilization in the final FOM analysis.
3. Power efficiency
The ranking of power efficiency was based on the required E_b/N_0 needed in order to achieve BER = 10^{-8}. This is the error rate requirement of the Constellation missions, but is otherwise arbitrary. Except for codes with known error floors, most notably the turbo codes, the particular choice of error rate requirement does not substantially affect the power efficiency ranking of the codes. This is because the codes surviving the first stage selection process are of roughly the same rate and length and are top-performing– thus, they have approximately the same slope in the waterfall region.
4. User burden
This is a measure of the cost to a mission of using a particular code. Codes of equivalent flight heritage whether extensive or nonexistent were assigned the same ranking.
5. Infrastructure burden
Codes already supported by the infrastructure (SN, GN, and DSN) were assigned higher ranking than those not supported by the infrastructure. Among those not already supported, the relative ranking of codes reflects the anticipated cost to implement their support.
6. Alignment with international standards
Codes in the CCSDS Blue Book were assigned the highest ranking. CCSDS Orange Book codes were assigned the next highest ranking. Codes that are IEEE, ITU, DVB or other standards for non-space applications were assigned the next highest ranking. The last ranking was used for codes that are not known to be part of any standard.
7. Robustness
With respect to coding, "robustness" captures (a) the ability of the code to operate in the presence of carrier synchronization error, symbol timing error, and non-AWGN noise, (b) the ability to detect when a decoder is unable to decode correctly (as opposed to putting out a decoded stream without knowing whether it is in error), and (c) the lack of an error floor at BER $< 10^{-8}$. As a general guide, codes surviving the first stage selection and having similar code rate and length have been observed to have similar performance with respect to (a). Therefore, the ranking was based primarily on (b) and (c).
8. Latency
As mentioned above, the first stage selection process has resulted in codes that are approximately the same length, and therefore, latency is given little additional weight in the fi-

nal FOM analysis. For block codes, the ranking of latency is based on the length of the block code. For the convolutional code, the ranking is based on the length of the traceback.

9. TRL

Codes with flight heritage are ranked the highest. Those with space technology demonstrations are assigned the next highest rank. Those with planned technology demonstrations are next, followed by those without any plans for space flight, and last, those without any known hardware implementation in the laboratory.

10. Capacity

For the purposes of comparison, the study assumes use of an AWGN channel, which is a good approximation for communications from space. The ranking of aggregate capacity, then, is a function of the E_b/N_0 required of the number of simultaneous links that are supported. Since the first stage selection procedure has resulted in codes of roughly the same code rate, the number of simultaneous links is the same for each candidate code, and the ranking reduces to a duplication of the power efficiency measure. As such, this FOM is given a low weight in the scoring system.

5. Final Code Selections

The initial code selections (first stage selection) identified the top codes for each link. In each case, we identified up to three candidate codes. For each link, we ranked the candidate codes using the FOM analysis of the previous section, and computed a weighted-average FOM rank.

Following step 7 of Section 3, we next identified the smallest set of recommended codes that would work well for all links scenarios. The FOM analysis indicates that for all links, CCSDS turbo and AR4JA LDPC codes uniformly outranked the turbo product codes and Flarion LDPC codes, and so they were eliminated. Among the remaining codes there was no smaller subset of codes that uniformly outperformed another code. This left uncoded, convolutional, turbo, AR4JA LDPC, and C_2 LDPC codes. Each of these codes had the best FOM score for at least one reference link, and so it would not be possible to reduce the set of recommended codes further without sacrificing link performance.

Tables 3 and 4 provide the recommended codes for Category A near-Earth SN/GN and Category B links, respectively. Although "International Standardization" is only one FOM, the final recommended links are in happy accordance with the CCSDS standards and ongoing working group activities. All the recommended codes are either contained in the CCSDS Blue Book (convolutional, turbo) or in an experimental CCSDS Orange Book (AR4JA LDPC, C_2 LDPC), as indicated in the color-coding of the tables.

6. Conclusions

The CMLP coding recommendations are the result of a long and detailed inter-center NASA study, and provides guidance and a schedule to the SCaN office for investing in the infrastructure needed to support the most powerful, cost-effective, means to meet NASA's communications needs in the coming decades. The final report of the study is also being reviewed by an international team, with the hope that we may have a unified approach to channel coding which will make future collaborative missions cost efficient and interoperable.

Appendices

A. Catalog of codes

The catalog of codes considered by the CMLP study is shown in Table 5. In addition to the columns shown, the study recorded the encoding latency, required E_b/N_0 with BPSK to achieve a given codeword error rate and undetected codeword error rate, standardization status, typical application, mission heritage, encoder/decoder hardware status, space qualification, speed, and commercial or military use.

B. Initial Code Selections

The initial code selection process detailed in section 3 resulted in selection of codes shown in Tables 6, 7, 8, and 9.

Acknowledgments

The research described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, the Goddard Space Flight Center, and by associated contracting companies, under a contract with the National Aeronautics and Space Administration.

References

[1] L. Deutsch, G. Noreen, J. Hamkins, J. Wesdock, F. Stocklin, and D. Zillig, "Selecting codes, modulations, multiple access schemes and link protocols for future nasa missions," in *IEEE Aerospace Conference*, Big Sky, Montana, Mar. 2008.

[2] "Space communication architecture working group (SCAWG) NASA space communication and navigation architecture recommendations for 2005-2030, final report," May 2006.

[3] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*. New Jersey: Prentice-Hall, 1983.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes," in *Proc., IEEE Int. Conf. on Communications*, May 1993, pp. 1064–1070.

[5] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *European Transactions on Telecommunications*, Mar. 1998.

[6] R. G. Gallager, "Low density parity check codes," *IRE Trans. Info. Theory*, vol. IT-8, pp. 21–28, 1962.

[7] F. Stocklin, L. Deutsch, G. Noreen, J. Hamkins, D. Lee,

Table 3. Final code selections, near Earth.

Link Description				Recommended Codes		
Direction	**Band**	**BW (MHz)**	**Data Rate (Mbps)**	**Code ID**	**Rate**	**Input length**
Forward (Uplink)	S-band	6	< 0.001	CC(7,1/2)	½	< 1000
			0.001 to 3	AR4JA LDPC	½	1024 to 16384
			3 to 4.8	AR4JA LDPC	2/3, 4/5	1024 to 16384
			> 4.8	C2 LDPC	0.87	7136
	X-band	50	< 25	AR4JA LDPC	½	1024 to 16384
Return (Downlink)	S-band	6	< 0.001	CC(7,1/2)	½	< 1000
			0.001 to 3	AR4JA LDPC	½	1024 to 16384
			3 to 4.8	AR4JA LDPC	2/3, 4/5	1024 to 16384
			> 4.8	C2 LDPC	0.87	7136
	S-band (launch)	20	16 to 22	AR4JA LDPC	½	1024 to 16384
	X-band	50	< 50	Turbo	1/6, 1/4, 1/3, 1/2	8920
			50 to 150	AR4JA & C2 LDPC	0.5 to 0.87	1024 to 16384
	Ka-band	650	< 300	Turbo	1/6, 1/4, 1/3, 1/2	8920
			300 to 650	AR4JA & C2 LDPC	1/2 to .87	1024 to 16384

Table 4. Final code selections, deep space.

Link Description				Recommended Codes		
Direction	**Band**	**BW (MHz)**	**Data Rate (Mbps)**	**Code ID**	**Rate**	**Input length**
Forward (Uplink)	X-band	50	< 0.001	CC(7,1/2)	½	< 1000
			0.001 to 40	AR4JA LDPC	1/2, 2/3, 4/5	1024 to 16384
			0.001 to 15	Turbo	1/6, 1/4, 1/3	1784 to 8920
			> 40	C2 LDPC	0.87	7136
	Ka-band	500	All	AR4JA LDPC	½	1024 to 16384
				Turbo	1/6, 1/4, 1/3	1784 to 8920
Return (Downlink)	X-band	50	< 50	Turbo	1/6, 1/4, 1/3, 1/2	8920
			50 to 150	AR4JA & C2 LDPC	0.5 to 0.87	1024 to 16384
	Ka-band	500	< 300	Turbo	1/6, 1/4, 1/3, 1/2	8920
			300 to 500	AR4JA & C2 LDPC	1/2 to .87	1024 to 16384

Table 5. Catalog of codes

	Code ID	r	k	n	**Required E_b/N_0 (dB) with BPSK for BER =**					
					10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-9}	10^{-10}
1	Uncoded	1.000	1	1	4.32	8.40	10.53	11.97	12.55	13.06
2	CC(3,1/2)	0.499	1022	2048	2.22	4.89	6.84	8.04	8.56	9.10
3	CC(5,1/2)	0.498	1020	2048	2.03	4.18	5.68	6.95	7.46	7.92
4	CC(7,1/2), delay=5 bits, Q=inf	0.500	Inf	Inf	3.40					
5	CC(7,1/2), delay=10 bits, Q=inf	0.500	Inf	Inf	2.77					
6	CC(7,1/2), delay=15 bits, Q=inf	0.500	Inf	Inf	2.39					
7	CC(7,1/2), delay=30 bits, Q=inf	0.500	Inf	Inf	1.88	3.51				
8	CC(7,1/2), delay=60 bits, Q=inf	0.499	1784	3574	1.70	3.40	4.78			
9	CC(7,1/2), delay=60 bits, Q=inf	0.500	3568	7142	1.70	3.40	4.78			
10	CC(7,1/2), delay=60 bits, Q=inf	0.500	8920	17846	1.70	3.40	4.78			
11	CC(7,1/2), delay=60 bits, Q=inf	0.500	16384	32774	1.70	3.40	4.78			
12	CC(7,1/2), delay=60 bits, Q=inf	0.500	Inf	Inf	1.70	3.40	4.78			
13	CC(7,1/2), delay=inf, hard dec.	0.500	Inf	Inf	3.67					
14	CC(7,1/2), delay=inf, Q=3	0.500	Inf	Inf	2.13	3.80	5.04	6.02	6.43	6.81
15	CC(7,1/2), delay=inf, Q=8	0.500	Inf	Inf	1.91	3.56				
16	CC(7,1/2), delay=inf, Q=inf	0.500	Inf	Inf	1.70	3.42				
17	CC(7,2/3), delay=60 bits, Q=inf	0.667	8920	13380	2.41	3.90	5.24			
18	CC(7,2/3), delay=120 bits, Q=inf	0.664	1024	1542	2.36	3.93	5.23	6.31	6.79	7.23
19	CC(7,3/4), delay=60 bits, Q=inf	0.750	8920	11894	2.93	4.47	5.77			
20	CC(7,3/4), delay=120 bits, Q=inf	0.747	1024	1371	2.84	4.42	5.78	6.91	7.40	7.84
21	CC(7,5/6), delay=60 bits, Q=inf	0.833	8920	10704	3.65	5.17	6.44			
22	CC(7,5/6), delay=120 bits, Q=inf	0.829	1024	1235	3.43	4.98	6.32	7.47	7.96	8.40
23	CC(7,7/8), delay=60 bits, Q=inf	0.875	8920	10195	4.20	5.83	7.25			
24	CC(7,7/8), delay=120 bits, Q=inf	0.871	1024	1176	3.90	5.36	6.68	7.90	8.44	8.93
25	CC(9,1/2), delay=45?, Q=inf	0.500	63	126	1.56	2.91	4.12	5.12	5.57	5.99
26	CC(9,1/2)	0.496	1016	2048	1.56	2.91	4.12	5.12	5.57	5.99
27	CC(9,1/2)	0.499	4088	8192	1.56	2.91	4.12	5.12	5.57	5.99
28	CC(15,1/4)	0.247	1010	4096	0.36	1.48	2.55	3.38	3.80	4.20
29	CC(15,1/6)	0.164	1010	6144	0.16	1.34	2.42	3.23	3.64	4.03
30	RS(255,223)	0.875	1784	2040	4.78	5.90	6.38	6.74	6.90	7.05
31	RS(255,239)	0.937	1912	2040	4.60	6.46	7.08	7.56	7.76	7.96
32	RS(252,220)	0.873	1760	2016	4.79	5.91	6.39	6.75	6.91	7.06
33	RS(255,223)+(7,1/2), I=1	0.437	1784	4080	1.97	2.56	2.94			
34	RS(255,223)+(7,1/2), I=2	0.437	3568	8160		2.34				
35	RS(255,223)+(7,1/2), I=3	0.437	5352	12240	1.89	2.27				
36	RS(255,223)+(7,1/2), I=4	0.437	7136	16320	1.89	2.24				
37	RS(255,223)+(7,1/2), I=5	0.437	8920	20400	1.88	2.23				
38	RS(255,223)+(7,1/2), I=8	0.437	14272	32640	1.88	2.19	2.40			
39	RS(255,223)+(7,1/2), I=16	0.437	28544	65280	1.89	2.20	2.39			
40	RS(255,239)+(7,1/2), I=1	0.469	1912	4080	1.84	2.72				
41	RS(255,239)+(7,1/2), I=2	0.469	3824	8160	1.82	2.47				
42	RS(255,239)+(7,1/2), I=3	0.469	5736	12240	1.82	2.37				
43	RS(255,239)+(7,1/2), I=4	0.469	7648	16320	1.83	2.33				
44	RS(255,239)+(7,1/2), I=5	0.469	9560	20400	1.85	2.32				
45	RS(255,239)+(7,1/2), I=8	0.469	15296	32640	1.83	2.30	2.58			
46	RS(255,239)+(7,1/2), I=16	0.469	30592	65280	1.84	2.30	2.59			
47	RS(255,223)+(7,1/2), I=5	0.469	4780	10200	1.88	2.23				
48	RS(255,223)+(7,2/3), I=5	0.625	6373	10200	2.62	2.92				
49	RS(255,223)+(7,3/4), I=5	0.703	7170	10200	3.15	3.47	3.68			
50	RS(255,223)+(7,5/6), I=5	0.781	7966	10200	3.87	4.22				
51	RS(255,223)+(7,7/8), I=5	0.820	8365	10200	4.39	4.77				
52	RS(255,239)+(7,1/2), I=5	0.469	4780	10200	1.85	2.32				

Table 5. Catalog of codes (continued)

	Code ID	r	k	n	**Required E_b/N_0 (dB) with BPSK for BER =**					
					10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-9}	10^{-10}
53	RS(255,239)+(7,2/3), I=5	0.625	6373	10200	2.54	2.97	3.25			
54	RS(255,239)+(7,3/4), I=5	0.703	7170	10200	3.06	3.52	3.79			
55	RS(255,239)+(7,5/6), I=5	0.781	7966	10200	3.79	4.27	4.56			
56	RS(255,239)+(7,7/8), I=5	0.820	8365	10200	4.33	4.83	5.15			
57	Turbo(1784,1/6)	0.166	1784	10728	-0.18	0.14	0.34	0.57		
58	Turbo(1784,1/4)	0.249	1784	7152		0.42	0.67			
59	Turbo(1784,1/3)	0.333	1784	5364	0.31	0.66	0.88			
60	Turbo(1784,1/2)	0.499	1784	3576	0.92	1.29	1.54	1.92		
61	Turbo(3568,1/6)	0.166	3568	21432		-0.04	0.14			
62	Turbo(3568,1/4)	0.250	3568	14288		0.23				
63	Turbo(3568,1/3)	0.333	3568	10716	0.22	0.47	0.62			
64	Turbo(3568,1/2)	0.499	3568	7144	0.83	1.10	1.27			
65	Turbo(7136,1/6)	0.167	7136	42840	-0.34	-0.17	-0.01			
66	Turbo(7136,1/4)	0.250	7136	28560	-0.08	0.11	0.26			
67	Turbo(7136,1/3)	0.333	7136	21420	0.16	0.33	0.45			
68	Turbo(7136,1/2)	0.500	7136	14280	0.78	0.97	1.11			
69	Turbo(8920,1/6)	0.167	8920	53544	-0.35	-0.20	-0.10	-0.02		
70	Turbo(8920,1/4)	0.250	8920	35696	-0.07	0.09	0.19	0.27	0.42	
71	Turbo(8920,1/3)	0.333	8920	26772	0.14	0.31	0.42	0.58		
72	Turbo(8920,1/2)	0.500	8920	17848	0.77	0.94	1.06	1.30		
73	Turbo(16384,1/6)	0.167	16384	98328						
74	Turbo(16384,1/4)	0.250	16384	65552						
75	Turbo(16384,1/3)	0.333	16384	49164						
76	Turbo(16384,1/2)	0.500	16384	32776						
77	BCH-SEC(63,56)	0.889	56	63	4.12	7.07	8.75	9.99	10.47	10.95
78	BCH-TED(63,56)	0.889	56	63	4.82	8.90	11.04	12.46	13.06	13.55
79	AR4JA(64,1/2)	0.500	64	128						
80	AR4JA(1024,1/2)	0.500	1024	2048	1.14	1.57	1.89	2.17	2.29	2.41
81	AR4JA(1024,2/3)	0.667	1024	1536	1.89	2.39	2.75	3.04	3.15	
82	AR4JA(1024,4/5)	0.800	1024	1280	2.77	3.36	3.76	4.14		
83	AR4JA(4096,1/2)	0.500	4096	8192	0.93	1.12	1.26	1.39		
84	AR4JA(4096,2/3)	0.667	4096	6144	1.67	1.90	2.07	2.20		
85	AR4JA(4096,4/5)	0.800	4096	5120	2.55	2.84	3.04	3.21		
86	AR4JA(16384,1/2)	0.500	16384	32768	0.75	0.87	0.96	1.03		
87	AR4JA(16384,2/3)	0.667	16384	24576	1.54	1.68	1.78			
88	AR4JA(16384,4/5)	0.800	16384	20480	2.47	2.64	2.74			
89	C_2, 50 iterations	0.875	7136	8160						4.19
90	TPC(128,120)^2	0.879	14400	16384	3.30	3.59	3.72	3.90		
91	TPC(64,57)^2	0.793	3249	4096	2.50	2.92	3.17	3.62		
92	TPC(53,46)×(51,44)	0.749	2024	2703	2.30	2.75	3.10	3.65		
93	TPC(39,32)^2	0.673	1024	1521	2.00	2.55	3.30	3.90		
94	TPC(32,26)^2	0.660	676	1024	1.80	2.50	3.15	3.85		
95	TPC(19,13)^2	0.468	169	361	1.65	2.85	4.10	5.00		
96	TPC(32,26)×(32,26)×(4,3)	0.495	2028	4096	1.50	1.90	2.40	3.10		
97	TPC(32,26)×(32,26)×(16,11)	0.454	7436	16384	1.30	1.46	1.58	1.72		
98	TPC(16,11)×(16,11)×(16,11)	0.325	1331	4096	0.90	1.21	1.50	1.80		
99	TPC(S16×H64^2)	0.744	48735	65536	2.30	2.50	2.55	2.60		
100	TPC(H64×H32×S32)	0.701	45942	65536	2.05	2.20	2.33	2.50		
101	TPC(H64×H32^2)	0.588	38532	65536	1.80	1.84	1.88	2.04		
102	TPC(H16×H64^2)	0.545	35739	65536	1.78	1.82	1.86	2.02		
103	TPC(S4×H16×H32^2)									
104	TPC(H16^4)									

Table 5. Catalog of codes (continued)

	Code ID	r	k	n	**Required E_b/N_0 (dB) with BPSK for BER =**					
					10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-9}	10^{-10}
105	TPC(S16×H32^2)	0.619	10140	16384	1.60	1.80	2.06	2.42		
106	BCH-LDPC(16200,1/4)	0.190	3072	16200			< 1.00			
107	BCH-LDPC(16200,1/3)	0.323	5232	16200			< 0.84			
108	BCH-LDPC(16200,2/5)	0.390	6312	16200			< 0.98			
109	BCH-LDPC(16200,1/2)	0.434	7032	16200			< 1.30			
110	BCH-LDPC(16200,3/5)	0.590	9552	16200			< 2.14			
111	BCH-LDPC(16200,2/3)	0.656	10632	16200			< 2.14			
112	BCH-LDPC(16200,3/4)	0.723	11712	16200			< 2.56			
113	BCH-LDPC(16200,4/5)	0.767	12432	16200			< 2.92			
114	BCH-LDPC(16200,5/6)	0.812	13152	16200			< 3.24			
115	BCH-LDPC(16200,8/9)	0.879	14232	16200			< 3.98			
116	BCH-LDPC(64800,1/4)	0.247	16008	64800			< 0.75			
117	BCH-LDPC(64800,1/3)	0.191	12408	64800			< 0.59			
118	BCH-LDPC(64800,2/5)	0.397	25728	64800			< 0.73			
119	BCH-LDPC(64800,1/2)	0.497	32208	64800			< 1.05			
120	BCH-LDPC(64800,3/5)	0.597	38688	64800			< 1.48			
121	BCH-LDPC(64800,2/3)	0.664	43040	64800			< 1.89			
122	BCH-LDPC(64800,3/4)	0.732	47408	64800			< 2.31			
123	BCH-LDPC(64800,4/5)	0.797	51648	64800			< 2.67			
124	BCH-LDPC(64800,5/6)	0.831	53840	64800			< 2.99			
125	BCH-LDPC(64800,8/9)	0.887	57472	64800			< 3.73			
126	BCH-LDPC(64800,9/10)	0.898	58192	64800			< 3.89			
127	Flarion- low threshold	0.500	4096	8192	0.72	0.95	1.11	>1.43		
128	Flarion- low floor	0.500	4096	8192	0.90	1.13	1.29	1.44	1.51	1.59
129		0.750	432	576						
130		0.750	1008	1344						
131		0.750	1728	2304						
132		0.500								
133		0.667								
134		0.833								
135	(3,4,7)LPDC(64)	0.500	64	128						
136	(3,4,7)LPDC(128)	0.500	128	256						
137	(3,4,7)LPDC(256)	0.500	256	512						
138										
139	F-LDPC (4096,)	0.500	4096	8192	1.25	1.58	1.78	1.95		
140	F-LDPC (4096, 2/3)	0.667	4096	6144	1.90	2.28	2.48	2.62		
141	F-LDPC (4096, 4/5)	0.800	4096	5120	2.75	3.13	3.36	3.52		
142	F-LDPC (4096, 8/9)	0.889	4096	4608	3.50	4.03	4.30	4.68		
143	F-LDPC (4096, 16/17)	0.941	4096	4352	4.10	4.98	5.32	5.90		
144	F-LDPC (8192,)	0.500	8192	16384	1.22	1.50	1.65	1.75		
145	F-LDPC (8192, 2/3)	0.667	8192	12288	1.90	2.18	2.34	2.48		
146	F-LDPC (8192, 4/5)	0.800	8192	10240	2.70	3.05	3.18	3.30		
147	F-LDPC (8192, 8/9)	0.889	8192	9216	3.50	3.90	4.10	4.25		
148	F-LDPC (8192, 16/17)	0.941	8192	8704	4.20	4.90	5.13	5.50		
149	F-LDPC(16k,)	0.500	16384	32768	1.10	1.25	1.32	1.40	1.44	1.48
150	F-LDPC(16k, 2/3)	0.667	16384	24576	1.80	1.94	2.02	2.11	2.15	2.18
151	F-LDPC(16k, 4/5)	0.800	16384	20480	2.65	2.83	2.93	3.03		
152	F-LDPC(16k, 8/9)	0.889	16384	18432	3.50	3.73	3.83	3.97		
153	F-LDPC(16k, 16/17)	0.941	16384	17408	4.20	4.68	4.82	5.00		
154	(3,1/2)+acc.									
155	CRC-32									
156	CRC-96									

Table 6. Initial code selections for deep space links.

Link Description				Recommended Modulation[1,2]			Link constraints, using recommended modulation				First Stage Recommended Codes[1]					
Direction	Band	Data Rate (Mbps)		Modulation ID	Shaping/Filter Type	Available BW (Mhz)	Spectral efficiency (b/s/Hz), 25 dB BW[3]	Used BW (Mhz), uncoded[4]	r lower bound[5]	k upper bound[6] based on 100 msec latency	Code ID	Code rate (r)	Input block length (k)	Latency (msec)	Bandwidth used (Mhz)	Required Eb/No to achieve BER=1e-8
		Min	Max													
Forward	X-band	0.001	0.001	PCM/PSK/PM	Unfiltered	4	0.05	0.02	0.006	100	Uncoded	1.000	1	1	0.0	12.00
											BCH-SEC(63,56)	0.889	56	56	0.0	9.99
											CC(7,1/2), delay=60 bits, Q=3	0.500	60	60	0.0	6.00
		0.001	0.128					2.84	0.710	100	Uncoded	1.000	1	1	2.8	12
											BCH-SEC(63,56)	0.889	56	56	3.2	9.99
											CC(7,1/2), delay=60 bits, Q=3	0.500	60	60	5.7	6
		0.128	8	OQPSK/PM	Butterworth 6th order	50	1.14	7.02	0.140	12800	Turbo(8920,1/6)	0.167	8920	70	42.1	-0.02
											Turbo(7136,1/6)	0.167	7136	56	42.1	0.19
		8	12					10.53	0.211	800000	Turbo(8920,1/4)	0.250	8920	1	42.1	0.27
											AR4JA(4096,1/2)	0.500	4096	1	21.1	1.39
											Turbo(1784,1/4)	0.249	1784	0	42.2	1
		12	16					14.04	0.281	1200000	Turbo(8920,1/3)	0.333	8920	1	42.1	0.58
											AR4JA(16384,1/2)	0.500	16384	1	28.1	1.03
											Flarion- low threshold	0.500	4096	0	28.1	1.43
		16	33					28.95	0.579	1600000	AR4JA(16384,2/3)	0.667	16384	1	43.4	1.85
											TPC(S16xH32^2)	0.619	10140	1	46.8	2.42
		33	40					35.09	0.702	3300000	TPC(H64xH32xS32)	0.701	45942	1	50.1	2.50
											AR4JA(16384,4/5)	0.800	16384	0	43.9	2.82
											AR4JA(4096,4/5)	0.800	4096	0	43.9	3.21
		40	50					43.86	0.877	4000000	C2, 50 iterations	0.875	7136	0	50.2	4.07
	Ka-band	1	75			500		65.79	0.132	100000	AR4JA(16384,1/2)	0.500	16384	16	131.6	1.03
		1	75								Turbo(8920,1/6)	0.167	8920	9	394.9	-0.02
		1	75								Turbo(7136,1/6)	0.167	7136	7	395.0	0.19
		75	250					219.3	0.439	7500000	AR4JA(16384,1/2)	0.500	16384	0	438.6	3.90
		250	500					438.6	0.877	25000000	C2, 50 iterations	0.875	7136	0	501.5	4.07
Return	X-band	0.001	0.001	PCM/PSK/PM	Unfiltered	4	0.05	0.02	0.006	100	Uncoded	1.000	1	1	0.0	12
		0.001	0.001								BCH-SEC(63,56)	0.889	56	56	0.0	9.99
		0.001	0.001								CC(7,1/2), delay=60 bits, Q=3	0.500	60	60	0.0	6
		0.001	0.128					2.84	0.710	100	Uncoded	1.000	1	1	2.8	12.00
		0.001	0.128								BCH-SEC(63,56)	0.889	56	56	3.2	9.99
		0.001	0.128								CC(7,1/2), delay=60 bits, Q=3	0.500	60	60	5.7	6.00
		0.128	6	Precoded GMSK (h = 0.5)	Gaussian (BT_b = 0.5)	50	0.77	7.79	0.156	12800	Turbo(8920,1/4)	0.250	8920	70	31.2	0.27
		0.128	6								AR4JA(4096,1/2)	0.500	4096	32	15.6	1.39
		0.128	6								Turbo(1784,1/4)	0.249	1784	14	31.2	1
		6	18	8-PSK	SRRC (BT_b = 0.35)		2.26	7.96	0.159	600000	Turbo(8920,1/6)	0.167	8920	1	47.8	-0.02
		6	18								Turbo(7136,1/6)	0.167	7136	1	47.8	0.19
		18	24	16-QAM	SRRC (BT_b = 0.35)		3.02	7.95	0.159	1800000	Turbo(8920,1/6)	0.167	8920	0	47.7	-0.02
		18	24								Turbo(7136,1/6)	0.167	7136	0	47.7	0.19
		24	50					16.56	0.331	2400000	Turbo(8920,1/3)	0.333	8920	0	49.7	0.58
		24	50								AR4JA(16384,1/2)	0.500	16384	1	33.1	1.03
		24	50								Flarion- low threshold	0.500	4096	0	33.1	1.43
		50	100					33.11	0.662	5000000	AR4JA(16384,2/3)	0.667	16384	0	49.7	1.85
		50	100								TPC(S16xH32^2)	0.619	10140	0	53.5	2.42
	Ka-band	1	75	Precoded GMSK (h = 0.5)	Gaussian (BT_b = 0.5)	500	0.77	97.4	0.195	100000	Turbo(8920,1/4)	0.250	8920	9	389.8	0.27
		1	75								AR4JA(4096,1/2)	0.500	4096	4	194.8	1.39
		1	75								Turbo(1784,1/4)	0.249	1784	2	390.5	1
		75	125					162.34	0.325	7500000	Turbo(8920,1/3)	0.333	8920	0	487.2	0.58
		75	125								AR4JA(16384,1/2)	0.500	16384	0	324.7	1.03
		125	250					324.68	0.649	12500000	AR4JA(16384,2/3)	0.667	16384	0	487.0	1.85
		125	250								TPC(S16xH32^2)	0.619	10140	0	524.6	2.42
		250	325					422.08	0.844	25000000	C2, 50 iterations	0.875	7136	0	482.6	4.07

Table 7. Initial code selections for near Earth, forward direction links.

Link Description				Recommended Modulation[1,2]			Link constraints, using recommended modulation				First Stage Recommended Codes[1]					
Direction	Band	Data Rate (Mbps)		Modulation ID	Shaping/Filter Type	Available BW (Mhz)	Spectral efficiency (b/s/Hz), 99% BW[3]	Used BW (Mhz), uncoded[4]	r lower bound[5]	k upper bound[6] based on 100 msec latency	Code ID	Code rate (r)	Input block length (k)	Latency (msec)	Bandwidth used (Mhz)	Required Eb/No to achieve BER=1e-8
		Min	Max													
Forward	S-band	0.018	0.1	Precoded GMSK (h = 0.5)	Gaussian (BT_b = 0.25)	6	1.16	0.09	0.014	1800	Turbo(1784,1/6)	0.166	1784	99	0.5	0.57
											TPC(16,11)x(16,11)x(16,11)	0.325	1331	74	0.3	1.80
											AR4JA(1024,1/2)	0.500	1024	57	0.2	2.17
		0.1	3					2.59	0.431	10000	Turbo(8920,1/2)	0.500	8920	89	5.2	1.3
											AR4JA(4096,1/2)	0.5	4096	41	5.2	1.39
		3	4.8					4.14	0.690	300000	TPC(H64xH32xS32)	0.701	45942	15	5.9	2.50
											AR4JA(16384,4/5)	0.800	16384	5	5.2	2.82
											AR4JA(4096,4/5)	0.800	4096	1	5.2	3.21
		4.8	6					5.17	0.862	480000	TPC(128,120)^2	0.879	14400	3	5.9	3.90
											F-LDPC(16k, 8/9)	0.889	16384	3	5.8	3.97
											C2, 50 iterations	0.875	7136	1	5.9	4.07
		0.018	0.1	OQPSK/PM	Butterworth 6th order		1.14	0.09	0.015	1800	Turbo(1784,1/6)	0.166	1784	99	0.5	0.57
											TPC(16,11)x(16,11)x(16,11)	0.325	1331	74	0.3	1.80
											AR4JA(1024,1/2)	0.500	1024	57	0.2	2.17
		0.1	3					2.63	0.439	10000	Turbo(8920,1/2)	0.500	8920	89	5.3	1.3
											AR4JA(4096,1/2)	0.5	4096	41	5.3	1.39
		3	4.8					4.21	0.702	300000	TPC(H64xH32xS32)	0.701	45942	15	6.0	2.50
											AR4JA(16384,4/5)	0.800	16384	5	5.3	2.82
											AR4JA(4096,4/5)	0.800	4096	1	5.3	3.21
		4.8	6					5.26	0.877	480000	TPC(128,120)^2	0.879	14400	3	6.0	3.90
											F-LDPC(16k, 8/9)	0.889	16384	3	5.9	3.97
											C2, 50 iterations	0.875	7136	1	6.0	4.07
	X-band	1	10	Precoded GMSK (h = 0.5)	Gaussian (BT_b = 0.25)	150	1.16	8.62	0.057	100000	Turbo(8920,1/4)	0.250	8920	9	34.5	0.27
											AR4JA(16384,1/2)	0.500	16384	16	17.2	1.03
											Turbo(1784,1/4)	0.249	1784	2	34.6	1
				OQPSK/PM	Butterworth 6th order		1.14	8.77	0.058	100000	Turbo(8920,1/4)	0.250	8920	9	35.1	0.27
											AR4JA(16384,1/2)	0.500	16384	16	17.5	1.03
											Turbo(1784,1/4)	0.249	1784	2	35.2	1

Table 8. Initial code selections for near Earth, return direction, S-Band links.

Link Description				Recommended Modulation[1,2]			Link constraints, using recommended modulation				First Stage Recommended Codes[1]					
Direction	Band	Data Rate (Mbps)		Modulation ID	Shaping/Filter Type	Available BW (Mhz)	Spectral efficiency (b/s/Hz), 99% BW[3]	Used BW (Mhz), uncoded[4]	r lower bound[5]	k upper bound[6] based on 100 msec latency	Code ID	Code rate (r)	Input block length (k)	Latency (msec)	Bandwidth used (Mhz)	Required Eb/No to achieve BER=1e-8
		Min	Max													
Return	S-band	0.018	0.1	Precoded GMSK (h = 0.5)	Gaussian (BT_b = 0.25)	6	1.16	0.09	0.014	1800	Turbo(1784,1/6)	0.166	1784	99	0.5	0.57
											TPC(16,11)x(16,11)x(16,11)	0.325	1331	74	0.3	1.80
											AR4JA(1024,1/2)	0.500	1024	57	0.2	2.17
		0.1	3					2.59	0.431	10000	Turbo(8920,1/2)	0.500	8920	89	5.2	1.3
											AR4JA(4096,1/2)	0.5	4096	41	5.2	1.39
		3	4.8					4.14	0.690	300000	TPC(H64xH32xS32)	0.701	45942	15	5.9	2.50
											AR4JA(16384,4/5)	0.800	16384	5	5.2	2.82
											AR4JA(4096,4/5)	0.800	4096	1	5.2	3.21
		4.8	6					5.17	0.862	480000	TPC(128,120)^2	0.879	14400	3	5.9	3.90
											F-LDPC(16k, 8/9)	0.889	16384	3	5.8	3.97
											C2, 50 iterations	0.875	7136	1	5.9	4.07
		0.018	0.1	OQPSK (SQPSK)	SRRC (roll-off factor = 0.5)		1.58	0.06	0.011	1800	Turbo(1784,1/6)	0.166	1784	99	0.4	0.57
											TPC(16,11)x(16,11)x(16,11)	0.325	1331	74	0.2	1.80
											AR4JA(1024,1/2)	0.500	1024	57	0.1	2.17
		0.1	3					1.9	0.316	10000	Turbo(8920,1/3)	0.333	8920	89	5.7	0.58
											AR4JA(4096,1/2)	0.500	4096	41	3.8	1.39
		3	4.8					3.04	0.506	300000	AR4JA(16384,1/2)	0.500	16384	5	6.1	1.03
											Turbo(8920,1/2)	0.500	8920	3	6.1	1.30
											AR4JA(4096,1/2)	0.500	4096	1	6.1	1.39
		4.8	6					3.8	0.633	480000	AR4JA(16384,2/3)	0.667	16384	3	5.7	1.85
											AR4JA(4096,2/3)	0.667	4096	1	5.7	2.20
											F-LDPC(16k, 2/3)	0.667	16384	3	5.7	2.11
		0.018	0.1	OQPSK/PM	Butterworth 6th order		1.14	0.09	0.015	1800	Turbo(1784,1/6)	0.166	1784	99	0.5	0.57
											TPC(16,11)x(16,11)x(16,11)	0.325	1331	74	0.3	1.80
											AR4JA(1024,1/2)	0.500	1024	57	0.2	2.17
		0.1	3					2.63	0.439	10000	Turbo(8920,1/2)	0.500	8920	89	5.3	1.3
											AR4JA(4096,1/2)	0.5	4096	41	5.3	1.39
		3	4.8					4.21	0.702	300000	TPC(H64xH32xS32)	0.701	45942	15	6.0	2.50
											AR4JA(16384,4/5)	0.800	16384	5	5.3	2.82
											AR4JA(4096,4/5)	0.800	4096	1	5.3	3.21
		4.8	6					5.26	0.877	480000	TPC(128,120)^2	0.879	14400	3	6.0	3.90
											F-LDPC(16k, 8/9)	0.889	16384	3	5.9	3.97
											C2, 50 iterations	0.875	7136	1	6.0	4.07
		16	22	8PSK	SRRC (roll-off factor = 0.35)	10	2.57	8.56	0.856	1600000	Uncoded	1	1	0	8.6	15.4
											C2, 50 iterations	0.875	7136	0	9.8	4.07

Table 9. Initial code selections for near Earth, return direction, X-Band and Ka-Band links.

Link Description				Recommended Modulation[1,2]			Link constraints, using recommended modulation				First Stage Recommended Codes[1]					
Direction	Band	Data Rate (Mbps)		Modulation ID	Shaping/Filter Type	Available BW (Mhz)	Spectral efficiency (b/s/Hz), 99% BW[3]	Used BW (Mhz), uncoded[4]	r lower bound[5]	k upper bound[6] based on 100 msec latency	Code ID	Code rate (r)	Input block length (k)	Latency (msec)	Bandwidth used (Mhz)	Required Eb/No to achieve BER=1e-8
		Min	Max													
Return	X-band	1	50	Precoded GMSK (h = 0.5)	Gaussian (BT_b = 0.25)	150	1.16	43.1	0.287	100000	Turbo(8920,1/3)	0.333	8920	9	129.4	0.58
											AR4JA(16384,1/2)	0.500	16384	16	86.2	1.03
		50	100					86.21	0.575	5000000	AR4JA(16384,2/3)	0.667	16384	0	129.3	1.85
											TPC(S16xH32^2)	0.619	10140	0	139.3	2.42
		100	150					129.31	0.862	10000000	TPC(128,120)^2	0.879	14400	0	147.1	3.90
											F-LDPC(16k, 8/9)	0.889	16384	0	145.5	3.97
											C2, 50 iterations	0.875	7136	0	147.9	4.07
		1	50	OQPSK/PM	Butterworth 6th order		1.14	43.86	0.292	100000	Turbo(8920,1/3)	0.333	8920	9	131.6	0.58
											AR4JA(16384,1/2)	0.500	16384	16	87.7	1.03
		50	100					87.72	0.585	5000000	AR4JA(16384,2/3)	0.667	16384	0	131.6	1.85
											TPC(S16xH32^2)	0.619	10140	0	141.7	2.42
		100	150					131.58	0.877	10000000	TPC(128,120)^2	0.879	14400	0	149.7	3.90
											F-LDPC(16k, 8/9)	0.889	16384	0	148.0	3.97
											C2, 50 iterations	0.875	7136	0	150.5	4.07
		1	50	OQPSK (SQPSK)	SRRC (roll-off factor = 0.5)		1.58	31.65	0.211	100000	Turbo(8920,1/4)	0.250	8920	9	126.6	0.27
											AR4JA(16384,1/2)	0.500	16384	16	63.3	1.03
		50	100					63.29	0.422	5000000	AR4JA(16384,1/2)	0.500	16384	0	126.6	1.03
											AR4JA(16384,2/3)	0.667	16384	0	94.9	1.85
		100	150					94.94	0.633	10000000	AR4JA(16384,2/3)	0.667	16384	0	142.4	1.85
											F-LDPC(16k, 2/3)	0.667	16384	0	142.4	2.11
											TPC(H64xH32xS32)	0.701	45942	0	135.4	2.50
		240	300	8PSK	SRRC (roll-off factor = 0.5)		2.37	126.58	0.844	24000000	TPC(128,120)^2	0.879	14400	0	144.0	3.90
											F-LDPC(16k, 8/9)	0.889	16384	0	142.4	3.97
											C2, 50 iterations	0.875	7136	0	144.7	4.07
					SRRC (roll-off factor = 0.35)		2.57	116.73	0.778		TPC(128,120)^2	0.879	14400	0	132.8	3.90
											F-LDPC(16k, 8/9)	0.889	16384	0	131.3	3.97
											C2, 50 iterations	0.875	7136	0	133.5	4.07
	Ka-band	1	650	Precoded GMSK (h = 0.5)	Gaussian (BT_b = 0.25)	650	1.16	560.34	0.862	100000	C2, 50 iterations	0.875	7136	7	640.8	4.07
				OQPSK (SQPSK)	SRRC (roll-off factor = 0.5)		1.58	411.39	0.633	100000	AR4JA(16384,2/3)	0.667	16384	16	617.1	1.85
				OQPSK/PM	Butterworth 6th order		1.14	570.18	0.877	100000	C2, 50 iterations	0.875	7136	7	652.0	4.07
		650	1000	OQPSK (SQPSK)	SRRC (roll-off factor = 0.2)		1.69	591.72	0.910	65000000	C2, 50 iterations	0.875	7136	0	676.6	4.07

J. Wesdock, and C. Patel, "Formulation of modulation recommendations for future nasa space communications," in *IEEE Aerospace Conference*, Big Sky, Montana, Mar. 2008.

BIOGRAPHY

Jon Hamkins received the B.S. degree in electrical engineering from the California Institute of Technology (Caltech), Pasadena, in 1990 and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 1993 and 1996, respectively. Since then, he has been with the Jet Propulsion Laboratory, Caltech, where he is now supervisor of the Information Processing Group.

Leslie Deutsch is the Chief Technologist and manager of the Architecture and Strategic Planning Office for the Interplanetary Networks Directorate at JPL. A mathematics graduate of Caltech, Dr. Deutsch has held various positions during his 27 years at JPL including management of several technology programs. Les was JPL's Chief Technologist for the 2002 fiscal year and developed JPL's strategy for technology development. Along the way Dr. Deutsch has published over 60 papers in the fields of communications, microelectronics, and spacecraft systems.

Dariush Divsalar received the Ph.D. dgree in electrical engineering from the University of California, Los Angeles in 1978. Since then, he has been with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, where he is a Principal Scientist.

Sam Dolinar received the S.B. degree in Physics, and S.M., E.E., and Ph.D. degrees in Electrical Engineering, from the Massachusetts Institute of Technology (MIT) in the 1970s. After four years at MIT Lincoln Laboratory, he has been at the Jet Propulsion Laboratory since 1980. His primary research interests are error correcting codes and data compression.

Dennis Lee earned his B.S. from Case Western Reserve University in 1997 and his M.S. from Rensselaer Polytechnic Institute in 1998, both in Electrical Engineering. Since 1999, he has been a member of technical staff in the Digital Signal Processing Research group at the Jet Propulsion Laboratory. Currently, his research interests include bandwidth efficient modulations and high rate signal processing.

Frank Stocklin joined NASA in 1967 as a test engineer for all Goddard Space Flight Center (GSFC) remote ground stations. He assumed responsibility for all link and geometric analysis for missions using the NASA Ground Network (GN) and Space Network (SN). He served as project engineer for the Gamma-Ray Remote TDRSS System installed at the Canberra Deep Space Network tracking station. He served as project engineer for the McMurdo TDRSS Relay System in Antartica (2 sites) and also for the TDRSS National Science Foundation South Pole station. Currently he is responsible for all RF/Geometric analyses for all missions using the GN/SN.

John Wesdock is a Program Manager at ITT Corporation, Advanced Engineering and Sciences division. Since joining ITT in 1990, John has provided extensive systems engineering support to the NASA/GSFC Space Network and Ground Network and their various customers. John currently oversees ITT's NASA Programs mathematical modeling and simulation group. John received his B.S.E.E. from the University of Pittsburgh in 1990 and his M.S.E.E. from George Washington University in 1994.

Chitra Patel is an RF Systems Engineer at ITT Industries Advanced Engineering and Sciences division. Since joining ITT in 1999, Chitra has primarily provided systems engineering support to the NASA/GSFC TDRSS program, most notably in the area of bandwidth efficient modulation and coding. Chitra received her B.S.E.E. from the Sardar Vallabhbhai Regional Engineering College in 1993 and her M.S.E.E. from George Washington University in 1999.