Modern Digital Satellite Television: How It Works

Matthew C. Valenti

Lane Department of Computer Science and Electrical Engineering
West Virginia University
U.S.A.

Nov, 1, 2011

Outline

Outline

Digital Satellite Television in the United States

DirecTV

- Spinoff of Hughes Network Systems.
- Began operations in 1994.
- 19.2 million U.S. subscribers at end of 2010.
- 23,000 employees in U.S. and Latin America.
- $ 33.6 billion market cap.

Dish Network.

- Spinoff of EchoStar.
- Began operations in 1996.
- 14.1 subscribers in 2010.
- 22,000 employees.
- $10.9 Billion market cap.

The DVB Family

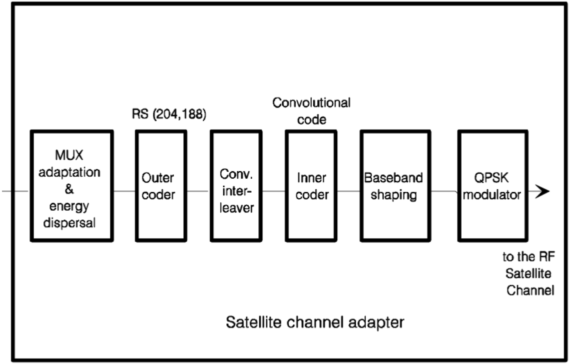
DVB is a family of open standards for digital video broadcasting.

- Maintained by 270-member industry consortium.
- Published by ETSI.

Modes of transmission

- Satellite: DVB-S, DVB-S2, and DVB-SH
- Cable: DVB-C, DVB-C2
- Terrestrial: DVB-T, DVB-T2, DVB-H

DVB-S



- Modulation: QPSK with $\alpha = 0.35$ rolloff.
- Channel coding: Concatenated Reed Solomon and convolutional.

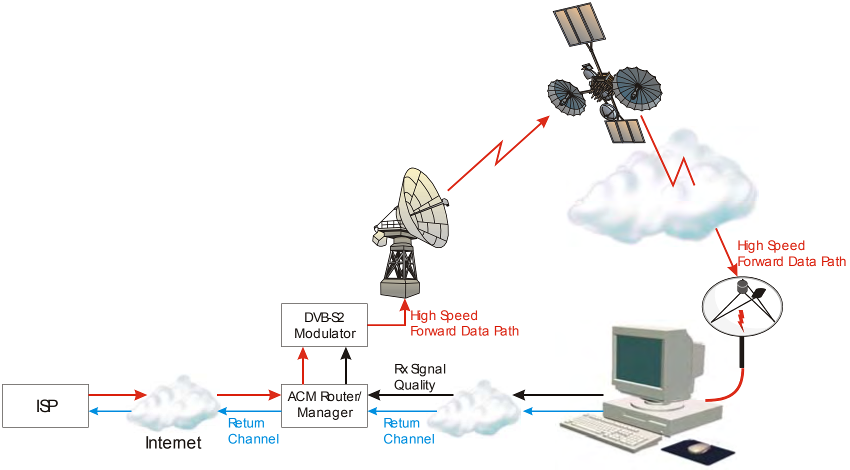
DVB-S2

DVB-S2 was introduced in 2003 with the following goals:

- Improve spectral efficiency by 30% through better modulation and coding.
  - Modulation: QPSK, 8PSK, 16/32 APSK.
  - Channel coding: LDPC with outer BCH code.
- Offer a more diverse range of services.
  - HDTV broadcast television.
  - Backhaul applications, e.g., electronic news gathering.
  - Internet downlink access.
  - Large-scale data content distribution, e.g., electronic newspapers.

Ratified 2005.

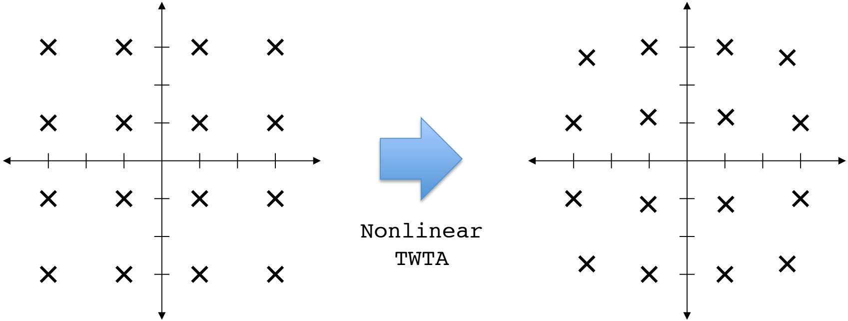
Adaptive Internet Downlink



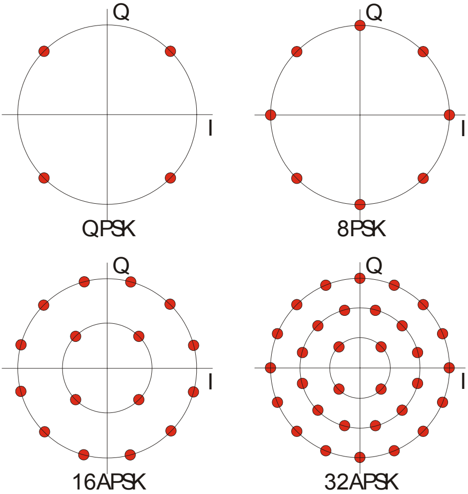
Outline

Why Not Use QAM?

- Higher spectral-efficiencies require larger signal constellations.
- However, nonlinear satellite channels are not well suited to square QAM.



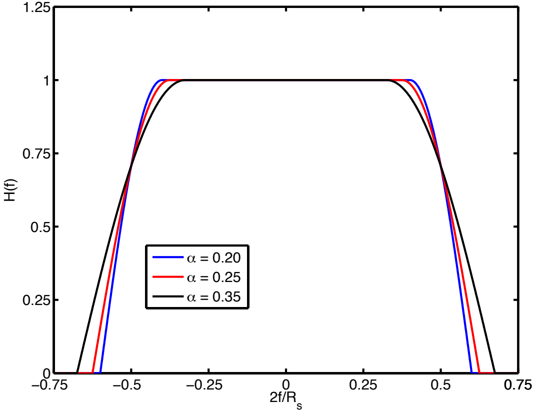
The DVB-S2 Signal Constellations



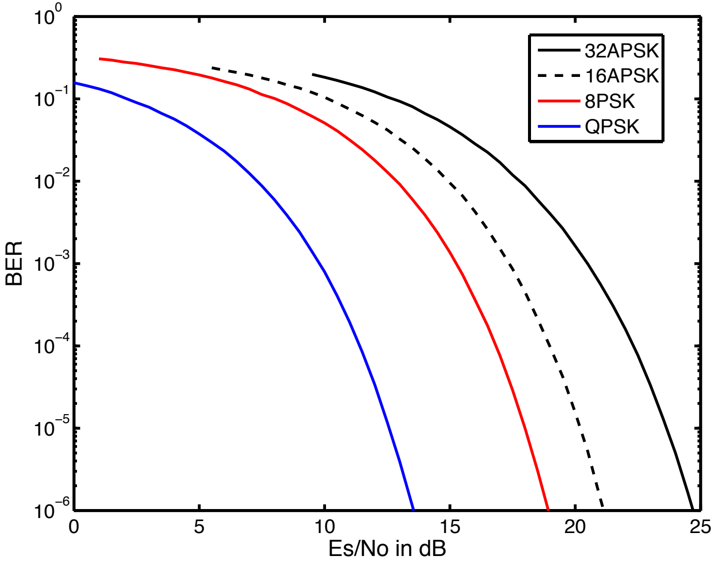
Raised-Cosine Rolloff Filtering

DVB-S2 uses a tighter root RC-rolloff filter.

- $B = R_s(1+\alpha)$
- Assuming a 6 MHz transponder channel...
- DVB-S Example:
  - $\alpha = 0.35$.
  - QPSK: $R_b = 2R_s$
  - $2(6)/(1.35) = 8.9$ Mbps
- DVB-S2 Example:
  - $\alpha = 0.20$.
  - 32-APSK: $R_b = 5R_s$
  - $5(6)/(1.2) = 25$ Mbps



BER in AWGN

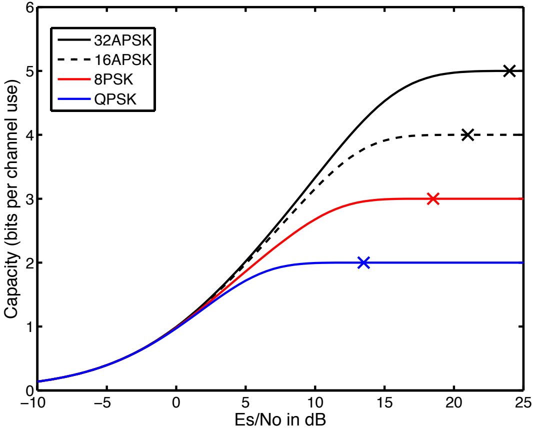


BER
Es/No in dB
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
5
10
15
20
25
32APSK
16APSK
8PSK
QPSK

Outline

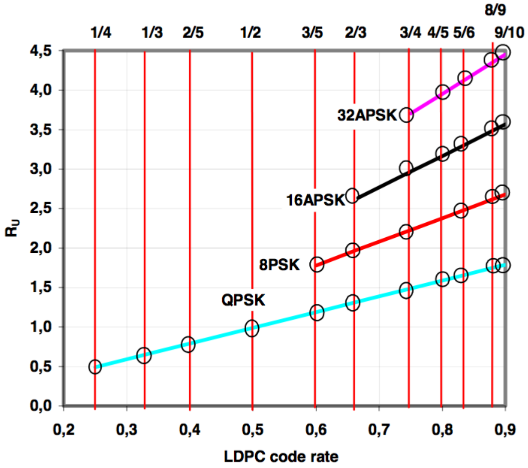
Capacity

- Performance can be improved by using error control coding.
- Gains are limited by the modulation-constrained capacity.
- LDPC codes are capable of approaching capacity.

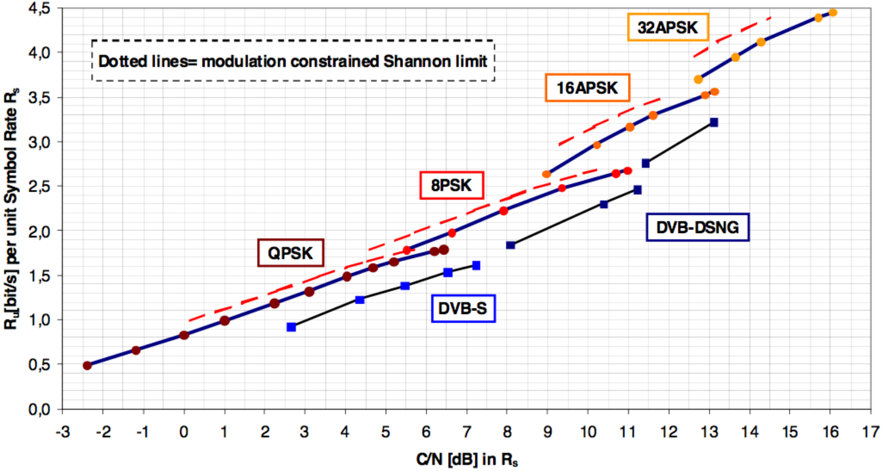


Available Code Rates

- The encoder maps length-k messages to length-n codewords.
- The code rate is $R = k/n$.
- Useful bit rate is $R_u = R \log_2(M)$.
- Two codeword lengths:
  - $16,200$.
  - $64,800$.



DVB-S2 vs. Shannon



Single Parity-Check Codes

- Consider the following rate $R = 5/6$ single parity-check code:

$$\mathbf{c} = [\underbrace{1 \quad 0 \quad 1 \quad 0 \quad 1}_{\mathbf{u}} \quad \underbrace{1}_{\text{parity bit}}]$$

- One error in *any* position may be detected:

$$\mathbf{c} = \begin{bmatrix} 1 & 0 & X & 0 & 1 & 1 \end{bmatrix}$$

- Problem with using an SPC is that it can only detect a single error.

Product Codes

- Place data into a k by k rectangular array.
  - Encode each row with a SPC.
  - Encode each column with a SPC.
  - Result is a rate $R = k^2/(k+1)^2$ code.
- Example $k = 2$.

$c_1 = u_1$	$c_2 = u_2$	$c_3 = c_1 \oplus c_2$		1	0	1
$c_4 = u_3$	$c_5 = u_4$	$c_6 = c_4 \oplus c_5$	$=$	1	1	0
$c_7 = c_1 \oplus c_4$	$c_8 = c_2 \oplus c_5$	$c_9 = c_3 \oplus c_6$		0	1	1

- A single error can be corrected by detecting its row and column location

1	0	1		1	0	1
0	1	0	$\Rightarrow$	1	1	0
0	1	1		0	1	1

Linear Codes

$c_1 = u_1$	$c_2 = u_2$	$c_3 = c_1 \oplus c_2$
$c_4 = u_3$	$c_5 = u_4$	$c_6 = c_4 \oplus c_5$
$c_7 = c_1 \oplus c_4$	$c_8 = c_2 \oplus c_5$	$c_9 = c_3 \oplus c_6$

- The example product code is characterized by the set of five linearly-independent equations:

$$\begin{aligned}
c_3 = c_1 \oplus c_2 &\quad\Rightarrow\quad c_1 \oplus c_2 \oplus c_3 = 0 \\
c_6 = c_4 \oplus c_5 &\quad\Rightarrow\quad c_4 \oplus c_5 \oplus c_6 = 0 \\
c_7 = c_1 \oplus c_4 &\quad\Rightarrow\quad c_1 \oplus c_4 \oplus c_7 = 0 \\
c_8 = c_2 \oplus c_5 &\quad\Rightarrow\quad c_2 \oplus c_4 \oplus c_8 = 0 \\
c_9 = c_3 \oplus c_6 &\quad\Rightarrow\quad c_3 \oplus c_6 \oplus c_9 = 0
\end{aligned}$$

- In general, it takes $(n - k)$ linearly-independent equations to specify a *linear* code.

Parity-check Matrices

- The system of equations may be expressed in matrix form as:

$$\mathbf{c}H^T = \mathbf{0}$$

where H is a *parity-check* matrix.

- Example:

$$\begin{array}{rcl} c_1 \oplus c_2 \oplus c_3 & = & 0 \\ c_4 \oplus c_5 \oplus c_6 & = & 0 \\ c_1 \oplus c_4 \oplus c_7 & = & 0 \\ c_2 \oplus c_4 \oplus c_8 & = & 0 \\ c_3 \oplus c_6 \oplus c_9 & = & 0 \end{array} \Leftrightarrow H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

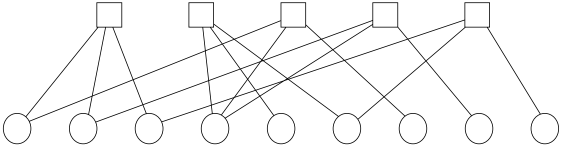
System of equations — Parity-check matrix

Tanner Graphs

- The parity-check matrix may be represented by a *Tanner* graph.
- Bipartite graph:
  - Check nodes: Represent the $n-k$ parity-check equations.
  - Variable nodes: Represent the n code bits.
- If $H_{i,j} = 1$, then i^{th} check node is connected to j^{th} variable node.
- Example: For the parity-check matrix:

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

The Tanner Graph is:



LDPC Codes

- Observations:
  - To achieve capacity, a long code is needed.
  - The decoder's complexity depends on the number of edges in the Tanner graph.
  - The number of edges is equal to the number of zeros in H.
  - It is desirable to have a code that is long, yet has a sparse H.
- Low-density parity-check codes:
  - An LDPC code is characterized by a *sparse* parity-check matrix.
  - The row/column weights are independent of length.
  - Decoder complexity grows only linearly with block length.
- Historical note:
  - LDPC codes were the subject of Robert Gallager's 1960 dissertation.
  - Were forgotten because the decoder could not be implemented.
  - Were "rediscovered" in the mid-1990's after turbo codes were developed.

Example LDPC Code

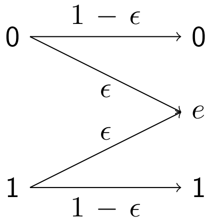
- A code from MacKay and Neal (1996):

$$\mathbf{H} = \left[\begin{array}{cccc|cccc|cccc}
1 & & & & & 1 & & 1 & & 1 & & \\
1 & & & 1 & 1 & & & & & & 1 & \\
 & 1 & & & 1 & & 1 & & 1 & & & \\
 & & 1 & & & 1 & & & & & 1 & 1 \\
 & & 1 & & & & 1 & 1 & & & & 1 \\
 & 1 & & & 1 & & & & 1 & & 1 & \\
1 & & & 1 & & & 1 & & & 1 & & \\
 & 1 & & & & 1 & & 1 & & 1 & & \\
 & & 1 & 1 & & & & & 1 & & & 1
\end{array}\right]$$

- The code is *regular* because:
  - The rows have constant weight (check-nodes constant degree).
  - The columns have constant weight (variable-nodes constant degree).
- This is called a $(3, 4)$ *regular* LDPC code because the variable nodes have degree 3 and the check nodes have degree 4.

The Binary Erasure Channel

- The binary erasure channel is a conceptual model used to explain the operation of LDPC codes.
- The BEC has two inputs (data 0 and data 1) and three outputs (data 0, data 1, and *erasure* e).
- A bit is erased with probability ϵ.
- A bit is correctly received with probability $1 - \epsilon$.



Erasures Decoding of Product Codes

- Several erasures may be corrected by iteratively decoding the SPC on each row and column.

e	e	1
1	e	0
0	e	e

Received word

$\Rightarrow$

e	e	1
1	1	0
0	e	e

Row decoding

$\Rightarrow$

1	e	1
1	1	0
0	e	1

Column decoding

$\Rightarrow$

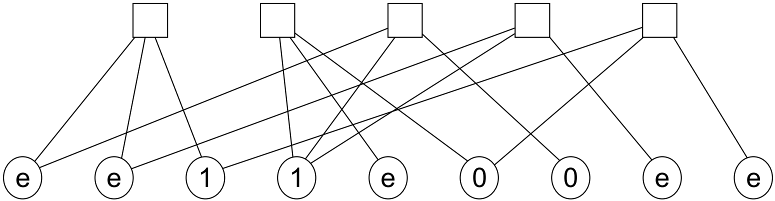
1	0	1
1	1	0
0	1	1

Row decoding

Erasures Decoding on the Tanner Graph

Decoding can be performed on the Tanner graph.

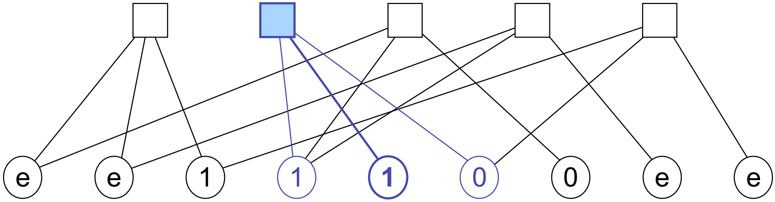
- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Erasures Decoding on the Tanner Graph

Decoding can be performed on the Tanner graph.

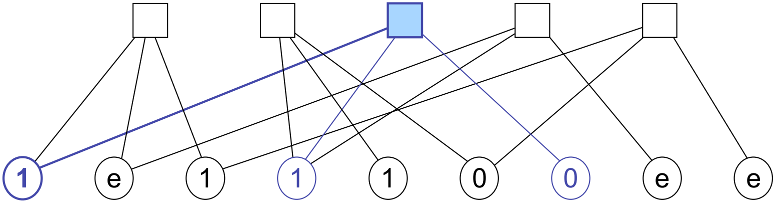
- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Erasures Decoding on the Tanner Graph

Decoding can be performed on the Tanner graph.

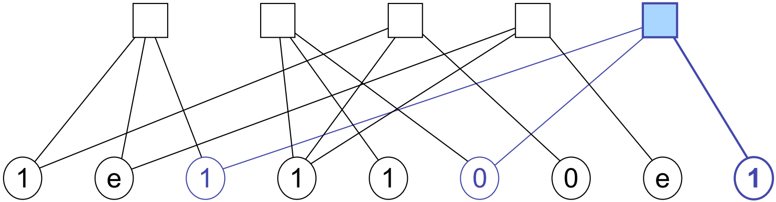
- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Erasures Decoding on the Tanner Graph

Decoding can be performed on the Tanner graph.

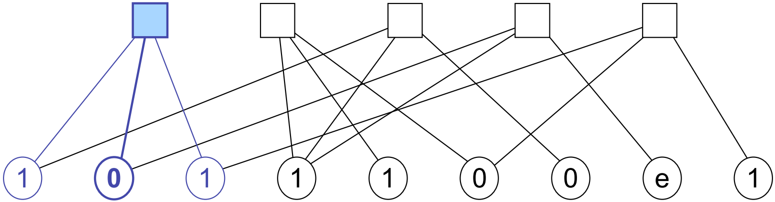
- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Erasures Decoding on the Tanner Graph

Decoding can be performed on the Tanner graph.

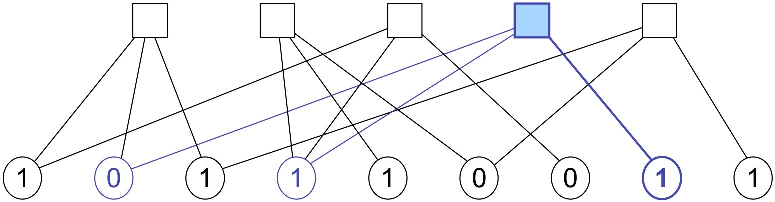
- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Erasures Decoding on the Tanner Graph

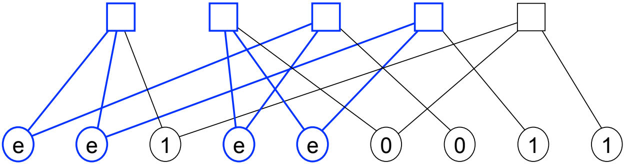
Decoding can be performed on the Tanner graph.

- Load the variable nodes with the observed code bits.
- Each check node j sends a *message* to each of its connected variable nodes i.
  - The message is the modulo two sum of the bits associated with the connected variable nodes *other* than i (if none are erased).
  - If a check node touches a *single* erasure, then it will become corrected.
- Iterate until all erasures corrected or no more corrections possible.



Stopping sets

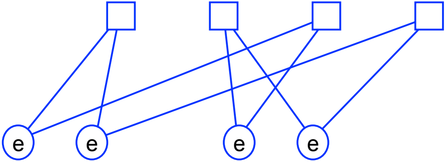
- A *stopping set* $\mathcal{V}$ is a set of erased variable nodes that cannot be corrected, regardless of the state of the other variable nodes.



- Let $\mathcal{G}$ be the neighbors of $\mathcal{V}$.
- Every check node in $\mathcal{G}$ touches at least two variable nodes in $\mathcal{V}$.
- The *minimimum* stopping set $\mathcal{V}_{min}$ is the stopping set containing the fewest variable nodes.
- Let $d_{min} = |\mathcal{V}_{min}|$ be the size of the minimum stopping set.
  - There exists at least one pattern of d_{min} erasures that cannot be corrected.
  - The erasure correcting capability of the code is $d_{min} - 1$, which is the maximum number of erasures that can always be corrected.

Stopping sets

- A *stopping set* $\mathcal{V}$ is a set of erased variable nodes that cannot be corrected, regardless of the state of the other variable nodes.



- Let $\mathcal{G}$ be the neighbors of $\mathcal{V}$.
- Every check node in $\mathcal{G}$ touches at least two variable nodes in $\mathcal{V}$.
- The *minimimum* stopping set $\mathcal{V}_{min}$ is the stopping set containing the fewest variable nodes.
- Let $d_{min} = |\mathcal{V}_{min}|$ be the size of the minimum stopping set.
  - There exists at least one pattern of d_{min} erasures that cannot be corrected.
  - The erasure correcting capability of the code is $d_{min} - 1$, which is the maximum number of erasures that can always be corrected.

Density Evolution

- For a (d_v, d_c) regular code, the probability that a variable-node remains erased after the ℓ^{th} iteration is

$$\epsilon_\ell = \epsilon_0 \left(1 - (1 - \epsilon_{\ell-1})^{d_c - 1}\right)^{d_v - 1} \quad (1)$$

  where d_v is the variable-node degree, d_c is the check-node degree, and the initial condition is $\epsilon_0 = \epsilon$.
- The above result assumes independent messages, which is achieved when the girth of the Tanner graph is sufficiently large.
- If $\epsilon_\ell \rightarrow 0$ as $\ell \rightarrow \infty$ for a particular channel erasure probability ϵ, then a code drawn from the ensemble of all such (d_v, d_c) regular LDPC codes will be able to correctly decode.
- The *threshold* ϵ^* is the maximum ϵ for which $\epsilon_\ell \rightarrow 0$ as $\ell \rightarrow \infty$.
- For the $(3, 6)$ regular code, the threshold is $\epsilon^* = 0.4294$

Proof of (1), Part I/II

- Decoding involves the exchange of messages between variable nodes and check nodes.
  - Let $p_\uparrow$ denote the probability of an erased message going *up* from the variable nodes to the check nodes.
  - Let $p_\downarrow$ denote the probability of an erased message going *down* from the check nodes to the variable nodes.
- Consider the degree d_c check node.
  - An outgoing message sent over a particular edge is a function of the incoming messages arriving over the other $d_c - 1$ edges.
  - For the outgoing message to be correct, all $d_c - 1$ incoming messages must be correct.
  - The outgoing message will be an erasure if any of the $d_c - 1$ incoming messages is an erasure.
  - The probability of the check node sending an erasure is:

$$p_\downarrow = 1 - (1 - p_\uparrow)^{d_c - 1} \qquad (2)$$

Proof of (1), Part II/II

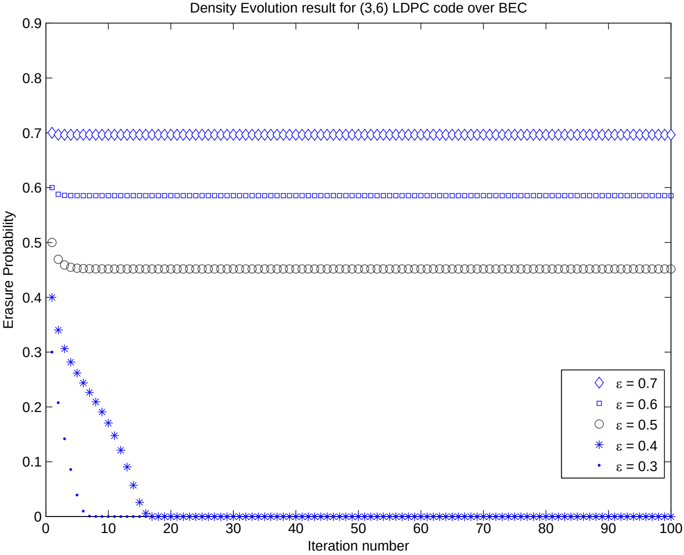
- Consider the degree d_v check node.
  - An outgoing message sent over a particular edge is a function of the incoming messages arriving over the other $d_v - 1$ edges.
  - An outgoing message will be an erasure if the variable node was initially erased *and* all of the arriving messages are erasures.
  - The probability of the variable node sending an erasure is:

$$p_\uparrow = \epsilon_0 p_\downarrow^{d_v - 1} \tag{3}$$

- Letting ϵ_ℓ equal the value of $p_\uparrow$ after the ℓ^{th} iteration, and substituting (2) into (3) yields the recursion given by (1):

$$\epsilon_\ell = \epsilon_0 \left(1 - (1 - \epsilon_{\ell-1})^{d_c - 1} \right)^{d_v - 1}$$

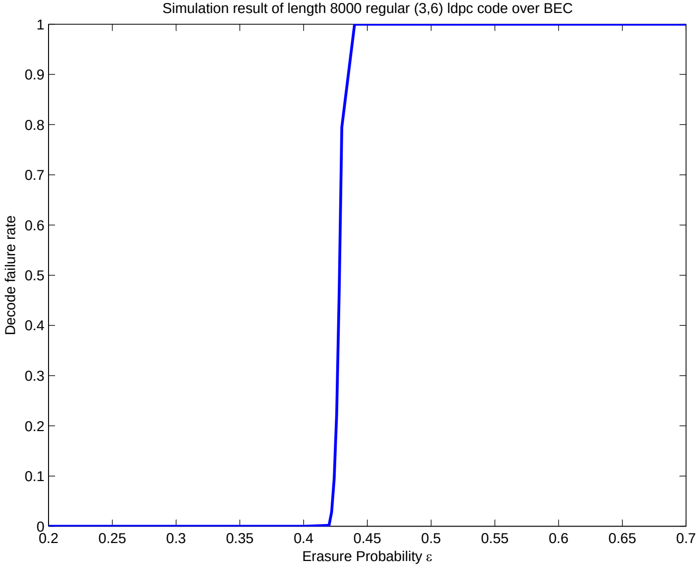
DE Example


Density Evolution result for (3,6) LDPC code over BEC
Erasure Probability
Iteration number
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
10
20
30
40
50
60
70
80
90
100
ε = 0.7
ε = 0.6
ε = 0.5
ε = 0.4
ε = 0.3

Code Realization

- Density evolution only describes the asymptotic performance of the *ensemble* of LDPC codes.
- Implementation requires that an H matrix be generated by drawing from the ensemble of all (d_v, d_c) LDPC codes.
- Goals of good H design:
    - High girth.
    - Full rank.
    - Large minimum stopping set.
- If the girth is too low, the short cycles invalidate the iterative decoder.
- High girth achieved through girth conditioning algorithms such as progressive edge growth (PEG).
- If H is not full rank, then the rate will be reduced according to the number of dependent equations.
- Small stopping sets give rise to an *error floor*.
- A database of good regular LDPC codes can be found on MacKay's website.

Performance of an Actual Code

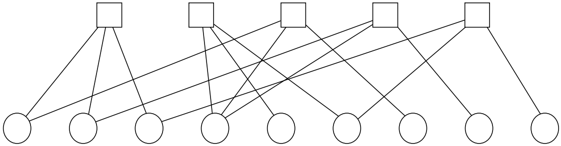


Irregular LDPC Codes

- Although regular LDPC codes perform well, they are not capable of achieving capacity.
- Properly designed irregular LDPC codes are capable of achieving capacity.
  - The degree distribution of the variable nodes is not constant.
  - The check-node degrees are often still constant (or close to it).
  - Here "designing" means picking the proper degree distribution.

Degree Distribution

- Edge-perspective degree distributions:
  - ρ_i is the fraction of *edges* touching degree i check nodes.
  - λ_i is the fraction of *edges* touching degree i variable nodes.
- For example, consider the Tanner graph:



  - 15 edges.
  - All are connected to degree-3 check nodes, so $\rho_3 = 15/15 = 1$.
  - Four are connected to degree-1 variable nodes, so $\lambda_1 = 4/15$.
  - Eight are connected to degree-2 variable nodes, so $\lambda_2 = 8/15$.
  - Three are connected to the degree-3 variable node, so $\lambda_3 = 3/15$.

DE for Irregular LDPC

- The degree distributions are described in polynomial form:
  - $\rho(x) = \sum_i \rho_i x^{i-1}$ for check nodes.
  - $\lambda(x) = \sum_i \lambda_i x^{i-1}$ for variable nodes.
- For an irregular code, the probability that a variable-node remains erased after the ℓ^{th} iteration is

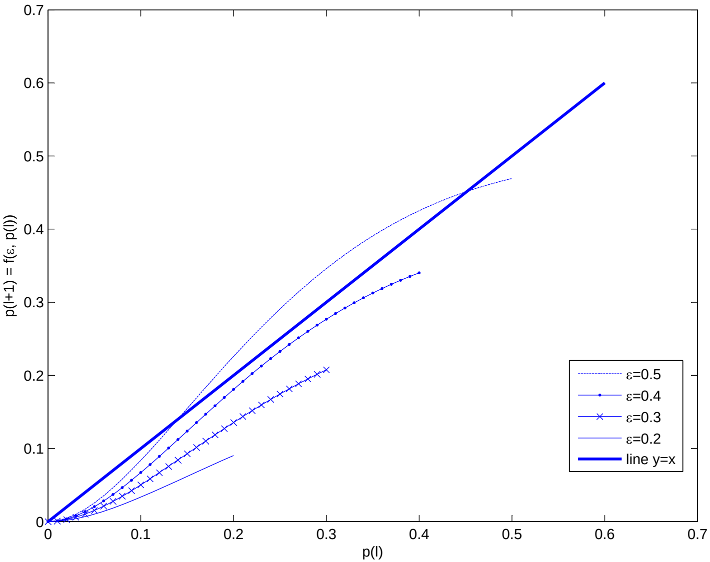
$$\epsilon_\ell = \epsilon_0 \lambda \left(1 - \rho \left(1 - \epsilon_{\ell-1}\right)\right)$$

  The proof follows from the Theorem on Total Probability.
- Convergence:
  - Error-free decoding requires that the erasure probability goes down from one iteration to the next.
  - Define the related function:

$$f(\epsilon, x) = \epsilon \lambda \left(1 - \rho \left(1 - x\right)\right)$$

  - Error-free decoding is possible iff $f(\epsilon, x) \leq x$ for all $0 \leq x \leq \epsilon$.

Convergence



Optimization

- The threshold is

$$\epsilon^* = \sup\{\epsilon : f(\epsilon, x) < x, \forall x, 0 < x \leq \epsilon\}$$

- Solving $f(\epsilon, x) = x$ for ϵ

$$\begin{aligned} x &= f(\epsilon, x) \\ &= \epsilon\lambda\left(1 - \rho\left(1 - x\right)\right) \\ \epsilon &= \frac{x}{\lambda\left(1 - \rho\left(1 - x\right)\right)} \end{aligned}$$

which is a function of x, and henceforth expressed as $\epsilon(x)$.

- This allows the threshold to be rewritten as:

$$\epsilon^* = \min\{\epsilon(x) : \epsilon(x) \geq x\}$$

Optimization with Linear Programming

- Our goal is to find the degree distribution which yields maximum threshold

$$\max_{\varepsilon^*} \{\varepsilon^* = \min(\varepsilon(x) : \varepsilon(x) \geq x)\};$$

- Several Constraints

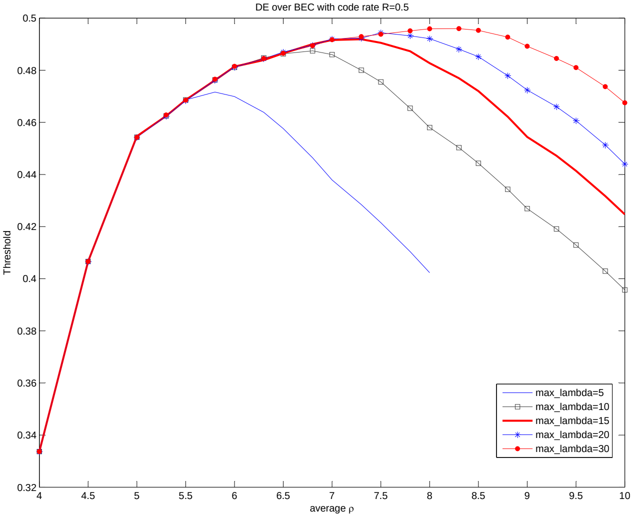
$$\frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} = 1 - R$$

$$\sum_{i \geq 2} \lambda_i = 1; \sum_{i \geq 2} \rho_i \quad = \quad 1;$$

$$x \in [0, 1]$$

- Which can be modeled as a optimization problem using linear programming
  - Can use Matlab's Optimization Toolbox.

Optimization Results ($\epsilon^* = 0.49596$)



Encoding LDPC Codes

Encoding of LDPC codes is not necessarily straightforward.

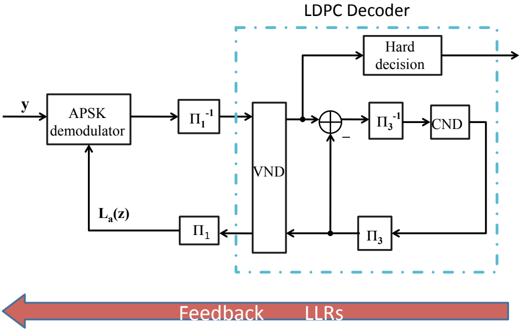
- "Systematic-form" H
  - Using Gaussian elimination, find $H = [P \; I]$.
  - Then $\mathbf{c} = \mathbf{u}G$ where $G = \left[I \; P^T\right]$.
  - However, P is likely to be high-density (complex encoding).
- Back-substitution.
  - If H is in an appropriate form, then $\mathbf{c}$ can be encoded using *back substitution*
  - Example, $\mathbf{c}H^T = \mathbf{0}$, where

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

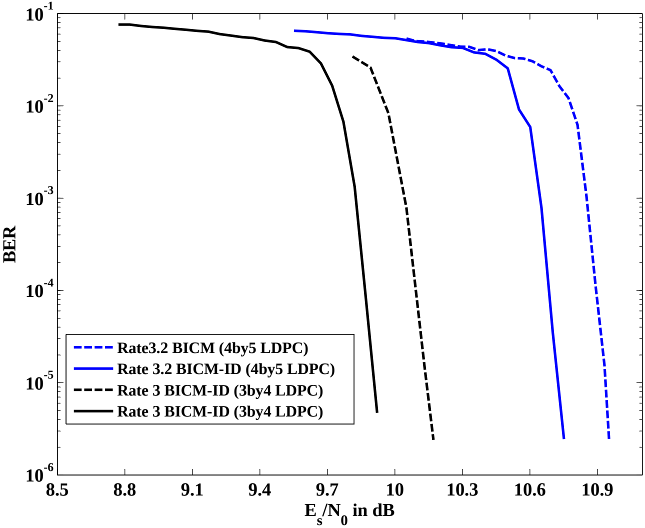
- The LDPC code in the DVB-S2 standard allows for back substitution.

Iterative Demodulation and Decoding

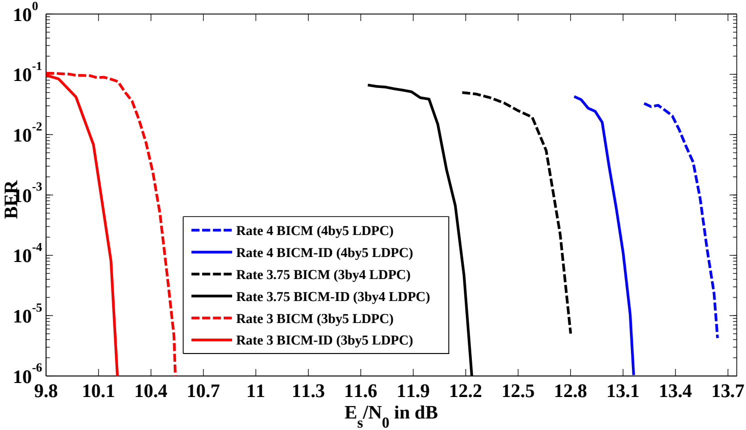
- Conventional receivers first demodulation, then decode.
- Performance is improved by iterating between the demodulator and decoder.
- BICM-ID: bit-interleaved modulation with iterative decoding.



AWGN Performance of 16APSK with BICM-ID



AWGN Performance of 32APSK with BICM-ID



Outline

Constellation Shaping

- Capacity curves assume equiprobable signaling.
- It is possible to increase capacity by transmitting higher-energy signals less frequently than lower-energy signals.

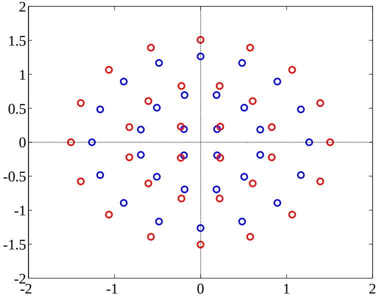


Figure: Uniform 32APSK o vs. shaped 32APSK o. Both constellations have the same energy.

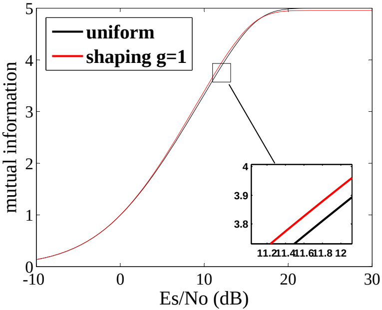


Figure: The capacity of shaped 32APSK is about 0.3 dB better than uniform 32APSK.

Sub-constellations

- The 32APSK is partitioned into two equal-sized sub-constellations.
- A *shaping bit* selects the sub-constellation, while the remaining bits select a symbol from the chosen sub-constellation.
- The lower-energy sub-constellation is selected more frequently.

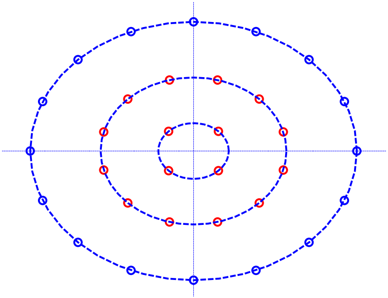


Figure: 32APSK w/ 2 sub-constellations

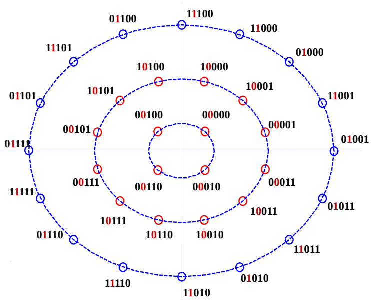


Figure: 32APSK symbol-labeling map

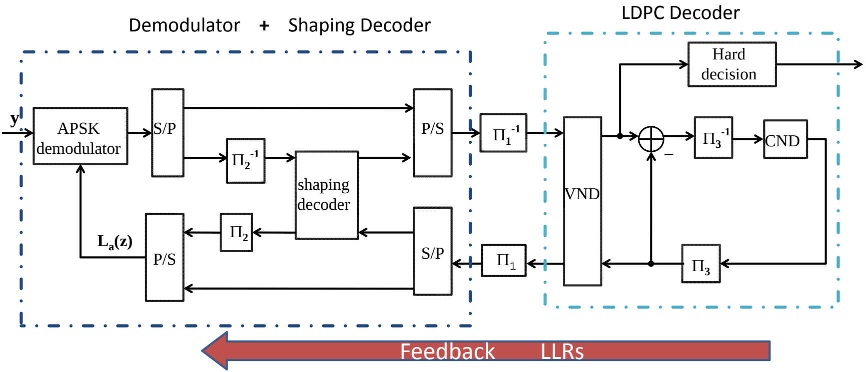
Shaping Encoder

- The shaping encoder should produce more zeros than ones.
- Example: $(n_s, k_s) = (5, 3)$

3 input data bits	5 output codeword bits
0 0 0	0 0 0 0 0
0 0 1	0 0 0 0 1
0 1 0	0 0 0 1 0
0 1 1	0 0 1 0 0
1 0 0	0 1 0 0 0
1 0 1	1 0 0 0 0
1 1 0	0 0 0 1 1
1 1 1	1 0 1 0 0

  - $p_0 = 31/40$: fraction of zeros.
  - $p_1 = 9/40$: fraction of ones.

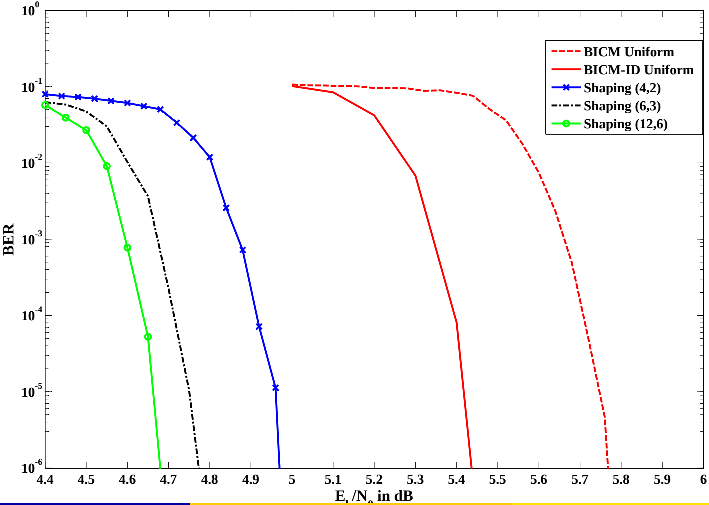
Receiver Implementation



- Additional complexity relative to BICM-ID due to shaping decoder.
- MAP shaping decoder compares against all 2^{k_s} shaping codewords.

BER of Shaping in AWGN

BER of 32-APSK in AWGN at rate R=3 bits/symbol



Outline

Conclusion

- DVB-S2 is a highly efficient system, thanks to
  - APSK modulation.
  - Tight RC-rolloff filtering.
  - Capacity-approaching LDPC codes.
- The performance of DVB-S2 can be improved by
  - BICM-ID.
  - Constellation shaping.
- Future work:
  - Using density evolution to optimize degree distribution of LDPC-coded APSK with shaping.
  - Extension to 64APSK and beyond.

Thank You.