Example

- Using k=3, rate 1/2 code from earlier slides
- Received: 11101100011000
- Some errors have occurred...
- What’s the 4-bit message?
- Look for message whose xmit bits are closest to rcvd bits

<table>
<thead>
<tr>
<th>Msg</th>
<th>Xmit*</th>
<th>Rcvd</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>000000000000</td>
<td>11101100011000</td>
<td>7</td>
</tr>
<tr>
<td>0001</td>
<td>0000001111110</td>
<td>11101100011000</td>
<td>8</td>
</tr>
<tr>
<td>0010</td>
<td>0000111110000</td>
<td>11101100011000</td>
<td>8</td>
</tr>
<tr>
<td>0011</td>
<td>0000101011110</td>
<td>11101100011000</td>
<td>4</td>
</tr>
<tr>
<td>0100</td>
<td>0011110000000</td>
<td>11101100011000</td>
<td>6</td>
</tr>
<tr>
<td>0101</td>
<td>0011110111110</td>
<td>11101100011000</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>0011001000000</td>
<td>11101100011000</td>
<td>7</td>
</tr>
<tr>
<td>0111</td>
<td>0011000101000</td>
<td>11101100011000</td>
<td>6</td>
</tr>
<tr>
<td>1000</td>
<td>1111100000000</td>
<td>11101100011000</td>
<td>4</td>
</tr>
<tr>
<td>1001</td>
<td>1111111111110</td>
<td>11101100011000</td>
<td>5</td>
</tr>
<tr>
<td>1010</td>
<td>1111011111110</td>
<td>11101100011000</td>
<td>7</td>
</tr>
<tr>
<td>1011</td>
<td>1111001000110</td>
<td>11101100011000</td>
<td>2</td>
</tr>
<tr>
<td>1100</td>
<td>1100010000000</td>
<td>11101100011000</td>
<td>5</td>
</tr>
<tr>
<td>1101</td>
<td>1100010111110</td>
<td>11101100011000</td>
<td>4</td>
</tr>
<tr>
<td>1110</td>
<td>1100100110000</td>
<td>11101100011000</td>
<td>6</td>
</tr>
<tr>
<td>1111</td>
<td>1100101001110</td>
<td>11101100011000</td>
<td>3</td>
</tr>
</tbody>
</table>

Most likely: 1011

*Msg padded with 2 zeros before transmission

Viterbi Algorithm

- Want: Most likely message sequence
- Have: (possibly corrupted) received parity sequences
- Viterbi algorithm for a given k and r:
  - Works incrementally to compute most likely message sequence
  - Uses two metrics
- Branch metric: BM(xmit,rcvd) measures likelihood that transmitter sent xmit given that we’ve received rcvd.
  - “Hard decision”: use digitized bits, compute Hamming distance between xmit and rcvd. Smaller distance is more likely if BER is small
  - “Soft decision”: use received voltages (more later...)
- Path metric: PM[s,i] for each state s of the $2^{k-1}$ transmitter states and bit time i where 0 ≤ i < len(message).
  - PM[s,i] = most likely BM(xmit,rcvd) over all message sequences m that leave transmitter in state s at time i
  - PM[s,i+1] computed from PM[s,i] and p_0[i],...,p_{r-1}[i]
Hard-decision Branch Metric

- BM = Hamming distance between expected parity bits and received parity bits
- Compute BM for each transition arc in trellis
  - Example: received parity = 00
    - BM(00,00) = 0
    - BM(01,00) = 1
    - BM(10,00) = 1
    - BM(11,00) = 2
- Will be used in computing PM[s,i+1] from PM[s,i].
- We’ll want most likely BM, which, since we’re using Hamming distance, means minimum BM.

Computing PM[s,i+1]

Starting point: we’ve computed PM[s,i], shown graphically as label in trellis box for each state at time i.

Example: PM[00,i] = 1 means there was 1 bit error detected when comparing received parity bits to what would have been transmitted when sending the most likely message, considering all messages that leave the transmitter in state 00 at time i.

Q: What’s the most likely state s for the transmitter at time i?  
A: state 00 (smallest PM[s,i])

Computing PM[s,i+1] cont’d.

Q: If the transmitter is in state s at time i+1, what state(s) could it have been in at time i?

A: For each state s, there are two predecessor states α and β in the trellis diagram

Example: for state 01, α=10 and β = 11.

Any message sequence that leaves the transmitter in state s at time i+1 must have left the transmitter in state α or state β at time i.

Computing PM[s,i+1] cont’d.

Example cont’d: to arrive in state 01 at time i+1, either
1) The transmitter was in state 10 at time i and the i\textsuperscript{th} message bit was a 0. If that’s the case, the transmitter sent 11 as the parity bits and there were 2 bit errors since we received 00. Total bit errors = PM[10,i] + 2 = 5  OR
2) The transmitter was in state 11 at time i and the i\textsuperscript{th} message bit was a 0. If that’s the case, the transmitter sent 01 as the parity bits and there was 1 bit error since we received 00. Total bit errors = PM[11,i] + 1 = 3

Which is mostly likely?
Computing PM\[s,i+1\] cont’d.

Formalizing the computation:

\[ PM[s,i+1] = \min(\text{PM}[\alpha,i] + \text{BM}[\alpha \rightarrow s], \text{PM}[\beta,i] + \text{BM}[\beta \rightarrow s]) \]

Example:

\[ PM[01,i+1] = \min(\text{PM}[10,i] + 2, \text{PM}[11,i] + 1) \]

\[ = \min(3+2,2+1) = 3 \]

Notes:
1) Remember with arc was min; saved arcs will form a path through trellis
2) If both arcs have same sum, break tie arbitrarily (eg, when computing PM[11,i+1])

Viterbi Algorithm

Finding the Most-likely Path

Example (cont’d.)

• Compute branch metrics for next set of parity bits
• Compute path metric for next column
  – add branch metric to path metric for old state
  – compare sums for paths arriving at new state
  – select path with smallest value (fewest errors, most likely)

• Path metric: number of errors on most-likely path to given state (min of all paths leading to state)
• Branch metric: for each arrow, the Hamming distance between received parity and expected parity

• After receiving 3 pairs of parity bits we can see that all ending states are equally likely
• Power of convolutional code: use future information to constrain choices about most likely events in the past
Survivor Paths

- Notice that some paths don’t continue past a certain state
  - Will not participate in finding most-likely path: eliminate
  - Remaining paths are called *survivor paths*
  - When there’s only one path: we’ve got a message bit!

Example (cont’d.)

- When there are “ties” (sum of metrics are the same)
  - Make an arbitrary choice about incoming path
  - If state is not on most-likely path: choice doesn’t matter
  - If state is on most-likely path: choice may matter and error correction has failed (mark state with underline to tell)

Example (cont’d.)

- When we reach end of received parity bits
  - Each state’s path metric indicates how many errors have happened on most-likely path to state
  - Most-likely final state has smallest path metric
  - Ties means end of message uncertain (but survivor paths may merge to a unique path earlier in message)

Traceback

- Use most-likely path to determine message bits
  - Trace back through path: message in reverse order
  - Message bit determined by high-order bit of each state (remember that came from message bit when encoding)
  - Message in example: 101100 (w/ 2 transmission errors)
Viterbi Algorithm Summary

- Branch metrics measure the likelihood by comparing received parity bits to possible transmitted parity bits computed from possible messages.

- Path metric $PM[s,i]$ measures the likelihood of the transmitter being in state $s$ at time $i$ assuming the mostly likely message of length $i$ that leaves the transmitter in state $s$.

- Most likely message? The one that produces the most likely $PM[s,N]$.

- At any given time there are $2^{k-1}$ most-likely messages we’re tracking $\Rightarrow$ time complexity of algorithm grows exponentially with constraint length $k$.

Soft Decisions

- Let’s limit the received voltage range to $[0.0,1.0]$  
  - $V_{eff} = \max(0.0, \min(1.0, V_{received})$  
  - Voltages outside this range are “good” 0’s or 1’s

- Define our “soft” branch metric as the square of the Euclidian distance between received $V_{eff}$ and expected voltages

- Soft-decision decoder chooses path that minimizes sum of the squares of the Euclidian distances between received and expected voltages  
  - Different branch metric but otherwise the same recipe

Hard Decisions

- As we receive each bit it’s immediately digitized to “0” or “1” by comparing it against a threshold voltage  
  - We lose the information about how “good” the bit is: a “1” at .9999V is treated the same as a “1” at .5001V

- The branch metric used in the Viterbi decoder is the Hamming distance between the digitized received voltages and the expected parity bits  
  - This is called hard-decision Viterbi decoding

- Throwing away information is (almost) never a good idea when making decisions  
  - Can we come up with a better branch metric that uses more information about the received voltages?

Channel Coding Summary

- Packetize  
- Block ECC  
- Interleave  
- Conv. Code  
- Bits to Volts  
- Volts to Bits  
- Viterbi  
- 10b8b  
- Deinterleave  
- Block ECC  
- Validate  
- Verify packet checksum

- Continuous coding

- Clock recovery, digitize “middle” sample

- Find most likely transmission

- Recover packets

- Spread burst errors into 1-bit errors

- Error correction using check bits

- Ensure sufficient transitions, packet syncs

- Build ECC blocks = data + check bits

- Order as 1st bits of all ECC blocks, 2nd bits,...