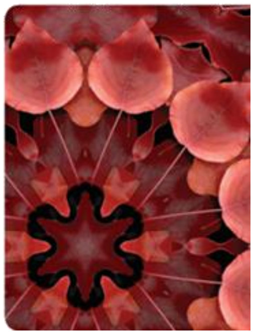
The Data Link Layer



1

Data Link Layer

- The Data Link Layer deals with algorithms for achieving:
  - Reliable, Efficient, communication of a whole units of information – frames (rather than the bits – Physical Layer) between two adjacent machines.
- Adjacent means that two machines are connected by a communication channel that acts conceptually like a wire (e.g., telephone line, coaxial cable, or wireless channel).
- Essential property of a channel that makes it "wire-like" connection is that the bits are delivered in exactly the same order in which they are sent.

2

Data Link Layer

- For ideal channel (no distortion, unlimited bandwidth and no delay) the job of data link layer would be trivial.
- However, limited bandwidth, distortions and delay makes this job very difficult.

3

Data Link Layer Design Issues

- Physical layer delivers bits of information to and from data link layer. The functions of Data Link Layer are:
  1. Providing a well-defined service interface to the network layer.
  2. Dealing with transmission errors.
  3. Regulating the flow of data so that slow receivers are not swamped by fast senders.

4

Data Link Layer Design Issues

- Network layer services
- Framing
- Error control
- Flow control

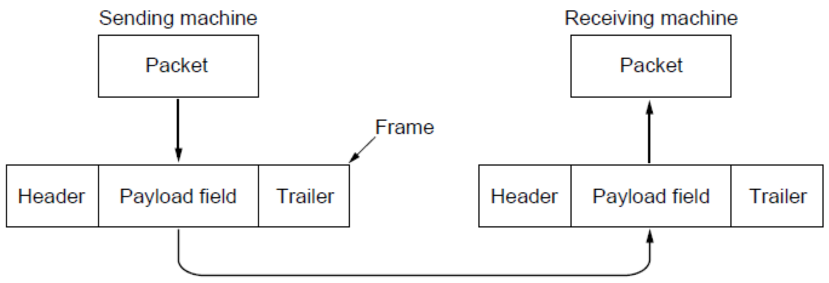
5

Data Link Layer Design Issues

- Data Link layer
  - Takes the packets from Network layer, and
  - Encapsulates them into **frames**
- Each frame has a
  - frame header
  - a payload field for holding the packet, and
  - frame trailer.
- Frame Management is what Data Link Layer does.

6

Packets and Frames



Relationship between packets and frames.

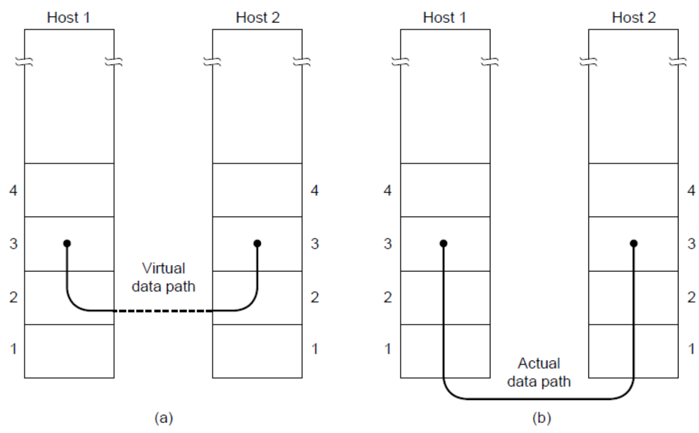
7

Services Provided to the Network Layer

- Principal Service Function of the data link layer is to transfer the data from the network layer on the source machine to the network layer on the destination machine.
  - On the source machine is an entity, call it a process in the network layer that hands some bits to the data link layer for transmission to the destination.
  - Job of data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer there

8

Network Layer Services



(a) Virtual communication. (b) Actual communication.

Possible Services Offered

- The data link layer can be designed to offer various services. The actual services that are offered vary from protocol to protocol. Three reasonable possibilities are:
  1. Unacknowledged connectionless service.
  2. Acknowledged connectionless service.
  3. Acknowledged connection-oriented service.

Unacknowledged Connectionless Service

- It consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them.
- Example: Ethernet, Voice over IP, etc., in all the communication channel where real time operation is more important than quality of transmission.

Acknowledged Connectionless Service

- When this service is offered, there are still no logical connections used.
- But, each frame send by the Data Link layer is acknowledged and the sender knows if a specific frame has been received or lost.
- Typically the protocol uses a specific time period that if has passed without getting acknowledgment it will re-send the frame.
- This service is useful for commutation when an unreliable channel is being utilized (e.g., 802.11 WiFi).
- It is perhaps worth emphasizing that providing acknowledgements in the data link layer is just an optimization, never a requirement.
- The network layer can always send a packet and wait for it to be acknowledged by its peer on the remote machine.
- Network layer does not know frame size of the packets and other restriction of the data link layer. Hence it becomes necessary for data link layer to have some mechanism to optimize the transmission.

Acknowledged Connection Oriented Service

- The most sophisticated service the data link layer can provide to the network layer is connection-oriented service.
- Source and Destination establish a connection first.
- Each frame sent is numbered
  - Data link layer guarantees that each frame sent is indeed received.
  - Furthermore, It guarantees that each frame is received only once and that all frames are received in the correct order.
  - It is appropriate over long, unreliable links.
- Examples:
  - Satellite channel communication,
  - Long-distance telephone communication, etc.

Acknowledged Connection Oriented Service

- When connection-oriented service is used, transfers go through three distinct phases:
  1. Connection is established by having both side initialize variables and counters needed to keep track of which frames have been received and which ones have not.
  2. One or more frames are transmitted.
  3. Finally, the connection is released – freeing up the variables, buffers, and other resources used to maintain the connection.

Framing

- To provide service to the network layer the data link layer must use the service provided to it by physical layer.
- Stream of data bits provided to data link layer is not guaranteed to be without errors.
- Errors could be:
  - Number of received bits does not match number of transmitted bits (deletion or insertion)
  - Bit Value
- It is up to data link layer to detect errors and if necessary correct the errors.

15

Framing

- Transmission of the data link layer starts with breaking up the bit stream
  - into discrete frames
  - Computation of a checksum for each frame, and
  - Include the checksum into the frame before it is transmitted.
- Receiver computes its checksum error for a receiving frame and if it is different from the checksum that is being transmitted will have to deal with the error.
- Framing is more difficult than one could think!
- A good design must make it easy for a receiver to find the start of new frames while using little of the channel bandwidth.

16

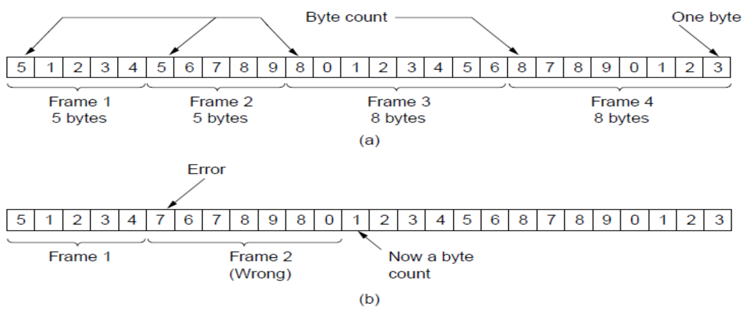
Framing Methods

1. Byte count.
2. Flag bytes with byte stuffing.
3. Flag bits with bit stuffing.
4. Physical layer coding violations.

17

Byte Count Framing Method

- It uses a field in the header to specify the number of bytes in the frame.
- Once the header information is being received it will be used to determine end of the frame.



A byte stream. (a) Without errors. (b) With one error.

18

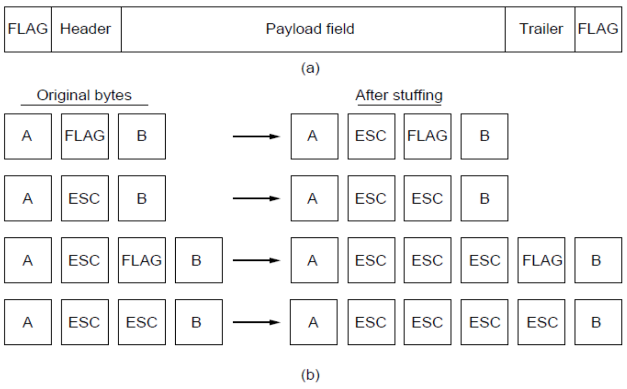
Byte Count Framing Method

- Trouble with this algorithm is that when the count is incorrectly received the destination will get out of synch with transmission.
  - Destination may be able to detect that the frame is in error but it does not have a means (in this algorithm) how to correct it.

Flag Bytes with Byte Stuffing Framing Method

- This methods gets around the boundary detection of the frame by having each appended by the frame start and frame end special bytes.
- If they are the same (beginning and ending byte in the frame) they are called **flag byte**.
- In the next slide figure this byte is shown as FLAG.
- If the actual data contains a byte that is identical to the FLAG byte (e.g., picture, data stream, etc.) the convention that can be used is to have escape character inserted just before the “FLAG” character.

Framing (2)



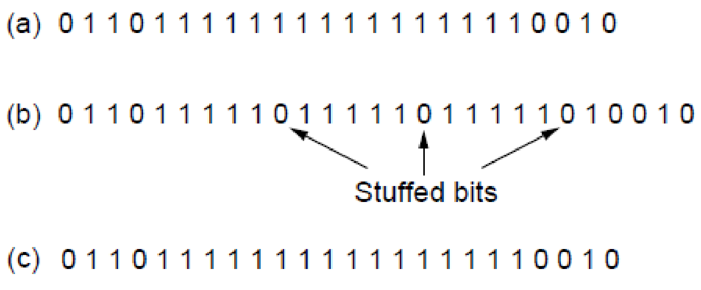
A frame delimited by flag bytes.

Four examples of byte sequences before and after byte stuffing.

Flag Bits with Bit Stuffing Framing Method

- This methods achieves the same thing as Byte Stuffing method by using Bit (1) instead of Byte (8 Bits).
- It was developed for High-level Data Link Control (HDLC) protocol.
- Each frames begins and ends with a special bit pattern:
  - 01111110 or 0x7E <- Flag Byte
  - Whenever the sender's data link layer encounters five consecutive 1s in the data it automatically stuffs a 0 bit into the outgoing bit stream.
  - USB uses bit stuffing.

Framing (3)


(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0
Stuffed bits
(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Bit stuffing. (a) The original data. (b) The data as they appear on the line. (c) The data as they are stored in the receiver's memory after destuffing.

Framing

- Many data link protocols use a combination of presented methods for safety. For example, in Ethernet and 802.11 each frame begin with a well-defined pattern called a preamble.
- Preamble is typically 72 bits long.
- It is then followed by a length field.

Error Control

- After solving the marking of the frame with start and end, the data link layer has to handle eventual errors in transmission.
  - Ensuring that all frames are delivered to the network layer at the destination are correct and in proper order.
- Unacknowledged connectionless service:
  - It is OK for the sender to output frames regardless of its proper reception.
- Reliable connection-oriented service:
  - It is NOT OK.

Error Control

- Reliable connection-oriented service usually will provide the sender with some feedback about what is happening at the other end of the line.
  - Receiver sends back special control frames.
  - If the Sender receives positive acknowledgment it will know that the frame has arrived safely.
- Timer and Frame Sequence Number for the sender is necessary to handle the case when there is no response (positive or negative) from the receiver .

Flow Control

- Important Design issue for the cases when the sender is running on a fast powerful computer and receiver is running on a slow low-end machine.
  - A common situation is when a smart phone requests a Web page from a far more powerful server, which then turns on the fire hose and blasts the data at the poor helpless phone until it is completely swamped.
  - Even if the transmission is error free, the receiver may be unable to handle the frames as fast as they arrive and will lose some.

Flow Control

- Two approaches:
  1. Feedback-based flow control
  2. Rate-based flow control
- Feedback-based flow control
  - Receiver sends back information to the sender giving it permission to send more data, or
  - Telling sender how receiver is doing.
- Rate-based flow control
  - Built in mechanism that limits the rate at which sender may transmit data, without the need for feedback from the receiver.

Error Detection and Correction

- Network designers have developed two basic strategies for dealing with errors. Both add redundant information to the data that is sent:
  1. Include enough redundant information to enable the receiver to deduce what the transmitted data must have been.
     Error correcting codes.
  2. Include only enough redundancy to allow the receiver to deduce that an error has occurred (but not which error)
     Error detecting codes.

  The use of error-correcting codes is often referred to as **FEC (Forward Error Correction).**

30

Error Detection and Correction

- Error codes are examined in Link Layer because this is the first place that we have run up against the problem of reliably transmitting groups of bits.
  - Codes are widely used because reliability is an overall concern.
  - The error correcting code are also seen in the physical layer for noisy channels.
  - Commonly they are used in link, network and transport layer.
- Error codes have been developed after long fundamental research conducted in mathematics.
- Many protocol standards get codes from the large field in mathematics.

31

Error Detection & Correction Code

- All the codes presented in previous slide add redundancy to the information that is being sent.
- A frame consists of
  - ***m*** data bits (message) and
  - ***r*** redundant bits (check).
- ***Block code*** - the ***r*** check bits are computed solely as function of the ***m*** data bits with which they are associated.
- ***Systemic code*** – the m data bits are sent directly, along with the check bits, rather than being encoded themselves before they are sent.
- ***Linear code*** – the r check bits are computed as a linear function of the m data bits. Exclusive OR (XOR) or modulo 2 addition is a popular choice. This means that encoding can be done with operations such as matrix multiplications or simple logic circuits.

Error Detection & Correction Code

- ***n*** – total length of a block (i.e., ***n*** = ***m*** + ***r***)
- (***n, m***) – code
- ***n*** – bit ***codeword*** containing m bits.
- ***m/n*** – code rate (range ½ for noisy channel and close to 1 for high-quality channel).

Error Detection & Correction Code

Example

- Transmitted: 10001001
- Received: 10110001

XOR operation gives number of bits that are different.

- XOR: 00111000
- Number of bit positions in which two codewords differ is called ***Hamming Distance***. It shows that two codes are d distance apart, and it will require d single bit errors to convert one into the other.
- Given the algorithm for computing the check bits, it is possible to construct a complete list of the legal codewords, and from this list to find the two codewords with the smallest Hamming distance. This distance is the Hamming distance of the complete code.

35

Error Detection & Correction Code

- All 2^m possible data messages are legal, but due to the way the check bits are computed not all 2^n possible code words are used.
- Only small fraction of $2^m/2^n=1/2^r$ of the possible messages will be legal codewords.
- The error-detecting and error-correcting capability of the block code depend on its Hamming distance.
- To reliably detect d error, one would need a distance $d+1$ code.
- To correct d error, one would need a distance $2d+1$ code.

36

Error Detection & Correction Code

Example:

- Consider a code with only four valid codewords:
  - 0000000000
  - 0000011111
  - 1111100000
  - 1111111111
- Minimal Distance of this code is 5 => can correct double errors and it can detect quadruple errors.
- If the codeword arrives 0000000111, and we expect only single or double bit error. Hence the receiving end must assume the original transmission was 0000011111.

37

Error Detection & Correction Code

- If a triple error changes 0000000000 in to 0000000111 , the error will not be corrected properly.
- Alternatively, if we expect all of these errors, we can detect them. None of the received codewords are legal codewords so an error must have occurred.
- One cannot correct double errors and at the same time detect quadruple errors.
- Error correction requires evaluation of each candidate codeword which may be time consuming search.
- Through proper design this search time can be minimized.

38

Hamming Code

- Lets assume that we want to design a code with *m* message bits and *r* check bits that will allow all single errors to be corrected.
- Each of the 2^m legal messages has *n* illegal codewords at a distance of 1 from it.
- These are formed by systematically inverting each of the *n* bits in the *n*-bit codeword formed from it.
- Thus, each of the 2^m legal messages requires *n* + 1 bit patterns dedicated to it. Since the total number of bit patterns is 2^n, we must have $(n + 1)2^m \leq 2^n$.
- Using n = m + r, this requirement becomes:
  - $(m + r + 1) \leq 2^r$
- Given m, this puts a lower limit on the number of check bits needed to correct single errors.

39

Hamming Code

- Suppose the message M is of m bits

m_m	-	-	-	-	m_3	m_2	m_1

- In Hamming code the Codeword will contain m+r bits, where r is the no.parity bits (or check bits):

	-	-	-	p_8	m_4	m_3	m_2	p_4	m_1	p_2	p_1
m+r	-	-	-	8	7	6	5	4	3	2	1

- Check bits: The bits that are powers of 2 (p_1, p_2, p_4, p_8, …).
- The rest of bits (m_1, m_2, m_3, m_4, m_5, …) are filled with ***m*** data bits.
- Example of the Hamming code with ***m*** = 7 data bits and ***r*** = 4 check bits is given in the next slide.

40

The Hamming Code

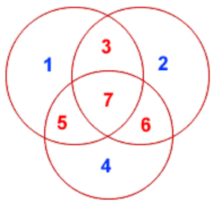
Consider a message having four data bits (D) which is to be transmitted as a 7-bit codeword by adding three error control bits. This would be called a (7,4) code. The three bits to be added are three EVEN Parity bits (P), where the parity of each is computed on different subsets of the message bits as shown below.

7	6	5	4	3	2	1	
D	D	D	P	D	P	P	7-BIT CODEWORD
D	-	D	-	D	-	P	(EVEN PARITY)
D	D	-	-	D	P	-	(EVEN PARITY)
D	D	D	P	-	-	-	(EVEN PARITY)

41

Hamming Code

- **Why Those Bits?** –
- The three parity bits (**1,2,4**) are related to the data bits (**3,5,6,7**) as shown in the figure below.
- In this figure, each overlapping circle corresponds to one parity bit and defines the four bits contributing to that parity computation.
- For example, data bit **3** contributes to parity bits **1** and **2**. Each circle (parity bit) encompasses a total of four bits, and each circle must have EVEN parity.
- Given four data bits, the three parity bits can easily be chosen to ensure this condition.



42

Hamming Code

- It can be observed that changing any one bit numbered 1..7 uniquely affects the three parity bits.
- Changing bit **7** affects all three parity bits, while an error in bit **6** affects only parity bits **2** and **4**, and an error in a parity bit affects only that bit.
- The location of any single bit error is determined directly upon checking the three parity circles.

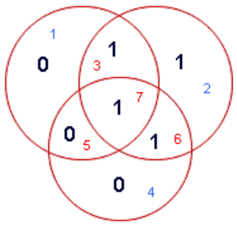
Hamming Code

- For example, the message 1101 would be sent as 1100110, since:

7	6	5	4	3	2	1	
1	1	0	0	1	1	0	7-BIT CODEWORD
1	-	0	-	1	-	**0**	(EVEN PARITY)
1	1	-	-	1	**1**	-	(EVEN PARITY)
1	1	0	**0**	-	-	-	(EVEN PARITY)

Hamming Codes

- When these seven bits are entered into the parity circles, it can be confirmed that the choice of these three parity bits ensures that the parity within each circle is EVEN, as shown here.



45

Hamming Code

- It may now be observed that if an error occurs in any of the seven bits, that error will affect different combinations of the three parity bits depending on the bit position.
- For example, suppose the above message 1100110 is sent and a single bit error occurs such that the codeword 1110110 is received:

```
transmitted message                    received message
       1 1 0 0 1 1 0     ------------>        1 1 1 0 1 1 0
BIT: 7 6 5 4 3 2 1                    BIT: 7 6 5 4 3 2 1
```

The above error (in bit 5) can be corrected by examining which of the three parity bits was affected by the bad bit:

46

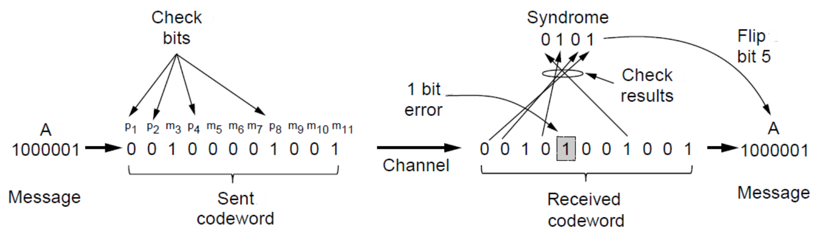
Hamming Code

7	6	5	4	3	2	1			
1	1	1	0	1	1	0	7-BIT CODEWORD		
1	-	1	-	1	-	0	(EVEN PARITY)	NOT!	1
1	1	-	-	1	1	-	(EVEN PARITY)	OK!	0
1	1	1	0	-	-	-	(EVEN PARITY)	NOT!	1

Hamming Code

- *In fact, the bad parity bits labeled* ***101*** *point directly to the bad bit since* ***101*** *binary equals* ***5****.* Examination of the 'parity circles' confirms that any single bit error could be corrected in this way.
- The value of the Hamming code can be summarized:
  1. Detection of 2 bit errors (assuming no correction is attempted);
  2. Correction of single bit errors;
  3. Cost of 3 bits added to a 4-bit message.
- The ability to correct single bit errors comes at a cost which is less than sending the entire message twice. (Recall that simply sending a message twice accomplishes no error correction.)

Hamming Code

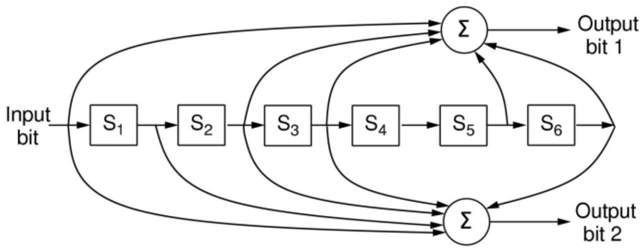


Example of an (11, 7) Hamming code correcting a single-bit error.

Convolutional Codes

- Not a block code
- There is no natural message size or encoding boundary as in a block code.
- The output depends on the current and previous input bits. Encoder has memory.
- The number of previous bits on which the output depends is called the **constraint length** of the code.
- They are deployed as part of the
  - GSM mobile phone system
  - Satellite Communications, and
  - 802.11 (see example in the previous slide).

Convolutional Codes



The NASA binary convolutional code used in 802.11.

51

Convolutional Encoders

- Like any error-correcting code, a convolutional code works by adding some structured redundant information to the user's data and then correcting errors using this information.
- A convolutional encoder is a *linear system*.
- A binary convolutional encoder can be represented as a *shift register*. The outputs of the encoder are modulo 2 sums of the values in the certain register's cells. The input to the encoder is either the unencoded sequence (for *non-recursive codes*) or the unencoded sequence added with the values of some register's cells (for *recursive codes*).
- Convolutional codes can be *systematic* and *non-systematic*. Systematic codes are those where an unencoded sequence is a part of the output sequence. Systematic codes are almost always recursive, conversely, non-recursive codes are almost always non-systematic.

52

Convolutional Encoders

- A combination of register's cells that forms one of the output streams (or that is added with the input stream for recursive codes) is defined by a *polynomial*. Let *m* be the maximum degree of the polynomials constituting a code, then $K=m+1$ is a *constraint length* of the code.

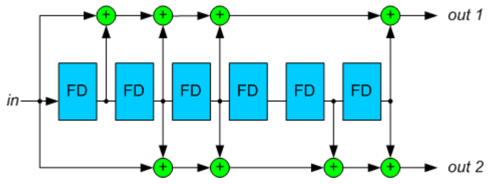


Figure 1. A standard NASA convolutional encoder with polynomials (171,133).

53

Convolutional Encoders

- For example, for the decoder on the Figure 1, the polynomials are:

$$g_1(z)=1+z+z^2+z^3+z^6$$
$$g_2(z)=1+z^2+z^3+z^5+z^6$$

- A code rate is an inverse number of output polynomials.
- For the sake of clarity, in this article we will restrict ourselves to the codes with rate $R=1/2$. Decoding procedure for other codes is similar.
- Encoder polynomials are usually denoted in the octal notation. For the above example, these designations are “1111001” = 171 and “1011011” = 133.
- The constraint length of this code is 7.
- An example of a recursive convolutional encoder is on the Figure 2.

54

Example of the Convolutional Encoder

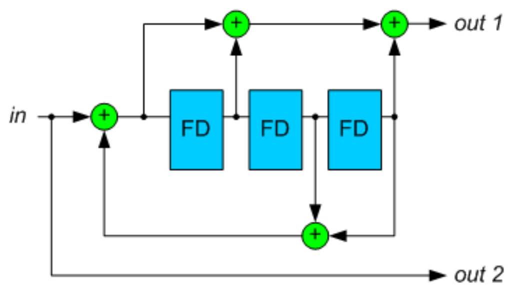


Figure 2. A recursive convolutional encoder.

55

Trellis Diagram

- A convolutional encoder is often seen as a *finite state machine*. Each state corresponds to some value of the encoder's register. Given the input bit value, from a certain state the encoder can move to two other states. These state transitions constitute a diagram which is called a *trellis diagram*.
- A trellis diagram for the code on the Figure 2 is depicted on the Figure 3. A solid line corresponds to input 0, a dotted line – to input 1 (note that encoder states are designated in such a way that the rightmost bit is the newest one).
- Each path on the trellis diagram corresponds to a valid sequence from the encoder's output. Conversely, any valid sequence from the encoder's output can be represented as a path on the trellis diagram. One of the possible paths is denoted as red (as an example).

56

Trellis Diagram

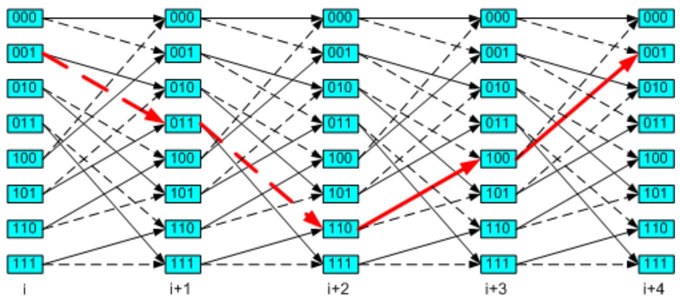


Figure 3. A trellis diagram corresponding to the encoder on the Figure 2.

Trellis Diagram

- Note that each state transition on the diagram corresponds to a pair of output bits.
- There are only two allowed transitions for every state, so there are two allowed pairs of output bits, and the two other pairs are forbidden.
- If an error occurs, it is very likely that the receiver will get a set of forbidden pairs, which don't constitute a path on the trellis diagram.
- So, the task of the decoder is to find a path on the trellis diagram which is the closest match to the received sequence.

Trellis Diagram

- Let's define a *free distance* d_f as a minimal Hamming distance between two different allowed binary sequences (a Hamming distance is defined as a number of differing bits).
- A free distance is an important property of the convolutional code. It influences a number of closely located errors the decoder is able to correct.

Viterbi Algorithm

- Viterbi algorithm reconstructs the maximum-likelihood path for a given input sequence.

Error-Detecting Codes (1)

- Error-correcting codes are widely used on wireless links, which are notoriously noisy and error prone when compared to optical fibers.
- However, over fiber or high- quality copper, the error rate is much lower, so error detection and retransmission is usually more efficient there for dealing with the occasional error.
- They are all linear, systematic block codes:
  1. Parity.
  2. Checksums.
  3. Cyclic Redundancy Checks (CRCs).

Parity Bit Error Detection

- In Parity Bit method, a single parity bit is appended to the data.
- The parity bit is chosen so that the number of 1 bits in the codeword is even (or odd).
- Doing this is equivalent to computing the (even) parity bit as the modulo 2 sum or XOR of the data bits.
  - For example, when 1011010 is sent in even parity, a bit is added to the end to make it 10110100.
  - With odd parity 1011010 becomes 10110101.

Parity Bit Error Detection

- Consider a channel on which errors are isolated and the error rate is 10^{-6} per bit.
- For Hamming Code of Block Size (m) 1000 bits, from the equation:

$$(m + r + 1) \leq 2^r$$

- We need (r) 10 check bits (1011 < 1024).
- 1 Mbit of data would require 10 kbits.
- To detect a block with a single bit of error, one parity bit would suffice.
- Once every 1000 blocks (bit error rate is 10^{-6}) one extra block would have to be re-transmitted.

63

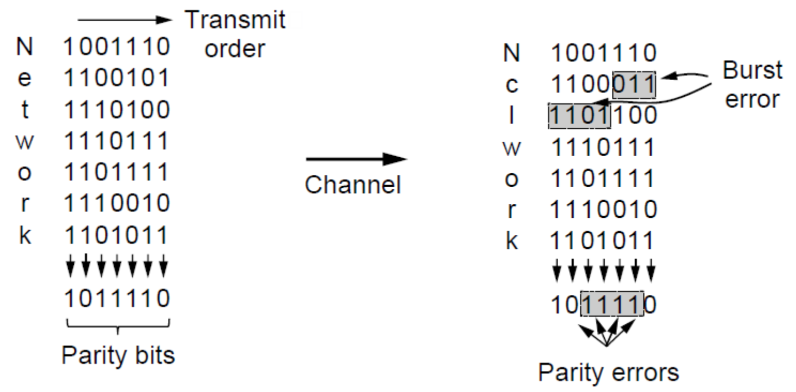
Parity Bit Error Detection

- One difficulty with this scheme is that a single parity bit can only reliably detect a single-bit error in the block.
- Problem with multiple bit errors in burst errors.
- The odds can be improved considerably if we consider the data stream as a matrix of k bits by n bits. Each (k) row is computed a parity.
- Up to *k* bit errors will be reliably detected as long as there is at most one error per row.
- **Interleaving** is used to compute parity in different order from the data that is being transmitted.
  - Compute parity for each n columns
  - Transmit the data as k rows.
  - The last row we will send the n parity bits.
- Example in the next slide gives the case for n=7 and k=7.

64

Parity Error-Detecting Codes (2)

Interleaving of parity bits to detect a burst error.



65

Parity Error-Detecting Codes

- A burst of length n+1 will pass undetected.
- The probability of that any of the n columns will have the correct parity by accident is 0.5; the probability of a bad block being accepted as a good one is 2^{-n}.

66

Checksum Error-Detecting Codes

- The second kind of error-detecting code, the checksum, is closely related to groups of parity bits.
- The word ‘‘checksum’’ is often used to mean a group of check bits associated with a message, regardless of how are calculated.
- Stronger checksums are based on a running sum of the data bits of the message.
- The checksum is usually placed at the end of the message, – as complementary sum.
  - Errors can be detected by summing the entire received codeword, both data bits and checksum bits.
  - If result is zero – no error has been detected.

Checksum Error-Detecting Codes

- Example of checksum is the 16-bit Internet error detection used as part of the IP protocol.
- It is applied to 16-bit words.
- It will detect an error for cases where parity detection fails.
- Checksum error would fail for:
  - Deletion or addition of zero data,
  - Swapping part of the message,
  - Messages splices in which parts of two packets are put together.
- Those are typical errors caused by faulty hardware.

Checksum Error-Detecting Codes

- Fletcher's checksum
  - Includes positional component- adding the product of the data and its position to the running sum.
  - This provides stronger detection of changes in the position of data.

CRC Error-Detecting Codes

- A third and stronger kind of error-detecting code is Cyclic Redundancy Check (**CRC**), also known as **polynomial code**.
- Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only.
- A k-bit frame is regarded as the coefficient list for a polynomial with k terms ranging from x^{k-1} to x^0
- Example:
  - 110001: $1x^5 + 1x^4 + 0x^3 + 0x^2 + 0x^1 + 1x^0$

CRC Error-Detecting Codes

- Polynomial arithmetic is done modulo 2 –
  - Addition and Subtraction are equivalent to exclusive OR.
  - Long division is carried the same way as in binary except that subtraction operation is again done module 2.
  - A divisor is said ‘‘to go into’’ a dividend if the dividend has as many bits as the divisor.

71

CRC Error-Detecting Codes

- Protocol requires that sender to agree in advance with the receiver on the **generator polynomial**, G(x).
  - Both high- and low-order bits of G(x) must be 1.
  - CRC is computed for a frame M(x) of length m- bits which is longer than the G(x).
  - The idea is to append a CRC to the end of the frame in such a way that the polynomial represented by the checksummed frame is divisible by G(x).
  - When the receiver gets a checksummed frame it divides it by G(x). If there the result is not equal to zero it means that there has been transmission error.

72

CRC Error-Detecting Codes

- Algorithm:
  1. Let r be the degree of G(x). Append r zero bits to the low-order end of the frame so it now contains m+r bits and corresponds to the polynomial $x^rM(x)$
  2. Divide the bit string corresponding to G(x) into the bit string corresponding to $x^rM(x)$ using modulo 2 division.
  3. Subtract the remanider (which is always r or fewer bits) from the bit string corresponding to $x^rM(x)$ using module 2 subtraction. The result is the checksummed frame, T(x), to be transmitted.

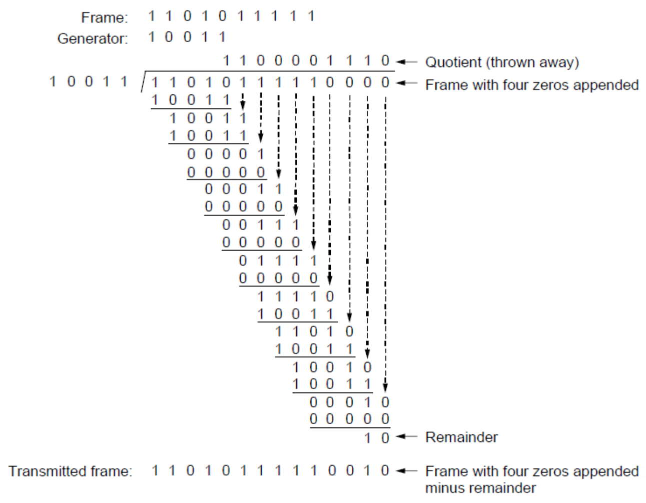
73

CRC Error-Detecting Codes

- Example:
  - Frame: 1101011111
  - Generator: x^4+x+1

74

Error-Detecting Codes (3)


Frame: 1 1 0 1 0 1 1 1 1 1
Generator: 1 0 0 1 1
1 1 0 0 0 0 1 1 1 0 Quotient (thrown away)
1 0 0 1 1 | 1 1 0 1 0 1 1 1 1 1 0 0 0 0 Frame with four zeros appended
1 0 Remainder
Transmitted frame: 1 1 0 1 0 1 1 1 1 1 0 0 1 0 Frame with four zeros appended minus remainder

Example calculation of the CRC

CRC Error-Detecting Codes

- Certain polynomials have become international standards.
- The one used in IEEE 802 followed the example of Ethernet and is:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{4} + x^{2} + x^{1} + 1$$

- Among other desirable properties, it has the property that it detects all bursts of length 32 or less and all bursts affecting an odd number of bits.
- Although the calculation required to compute the CRC may seem complicated, it is easy to compute and verify CRCs in hardware with simple shift register circuits.