3. Data Link Layer

Rakesh Matam

Indian Institute of Information Technology Guwahati

rakesh@iiitg.ac.in

August 13, 2021

Objectives

- Understand the design principles of DLL algorithms for achieving reliable, efficient communication of whole units of information called frames between two **adjacent machines**.
- Error correction and Flow Control.

DLL Design Issues

The data link layer uses the services of the physical layer to send and receive bits over communication channels. It has a number of functions, including:

- Providing a well-defined service interface to the network layer.
- Dealing with transmission errors.
- Regulating the flow of data so that slow receivers are not swamped by fast senders.

DLL Design Issues

The data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission.

Each frame contains a frame header, a payload field for holding the packet, and a frame trailer.

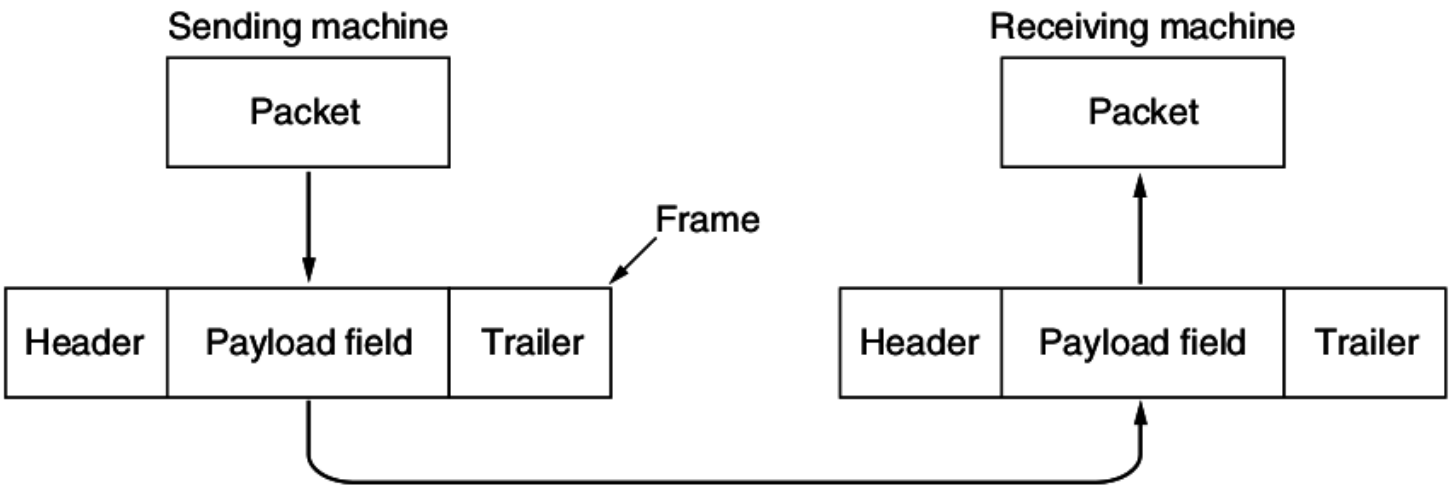


Figure 3-1. Relationship between packets and frames.

Services Provided to the Network Layer

The principal service is transferring data from the network layer on the source machine to the network layer on the destination machine.

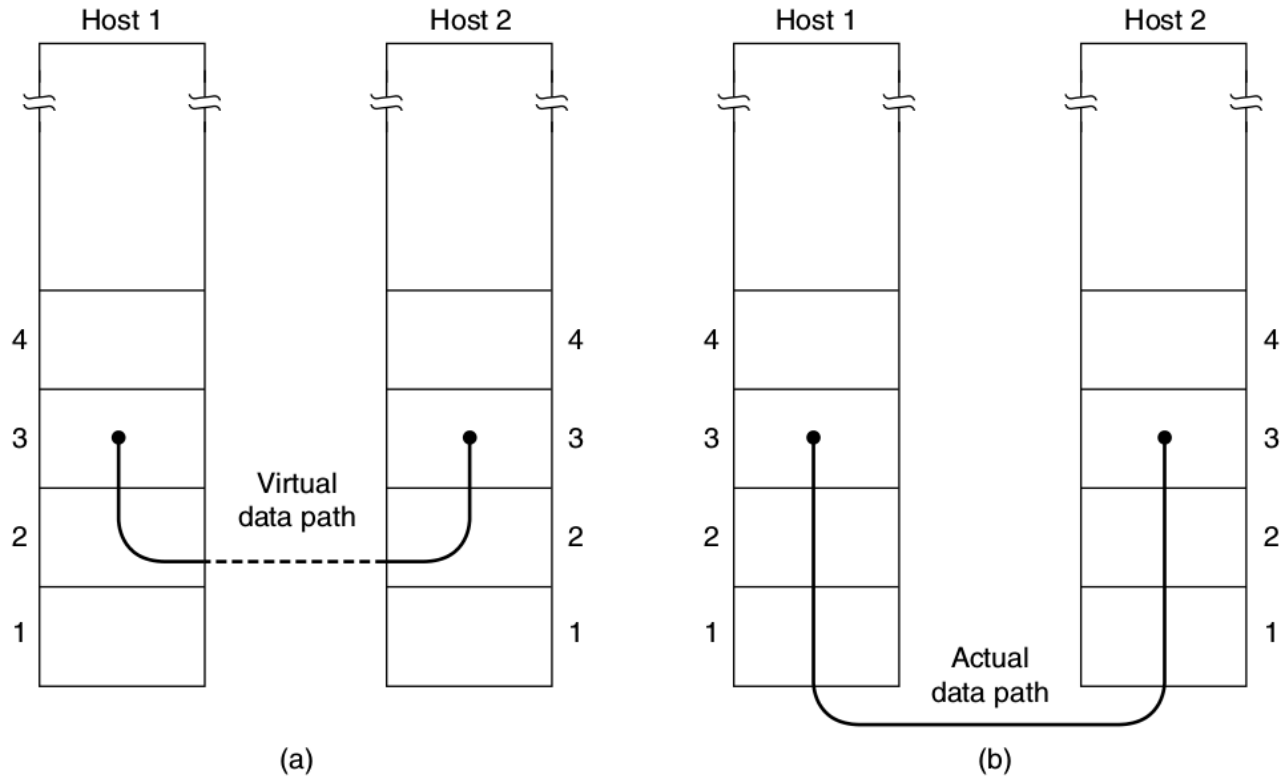


Figure 3-2. (a) Virtual communication. (b) Actual communication.

Types of Services

Unacknowledged connectionless service

- Frame is sent with no connection / error recovery
- Ethernet is example

Acknowledged connectionless service

- Frame is sent with retransmissions if needed
- Example is 802.11

Acknowledged connection-oriented

- connection oriented service
- Connection is set up; rare

Framing

To provide service to the network layer, the data link layer must use the service provided to it by the physical layer.

The physical layer simply accepts a raw bit stream and attempt to deliver it to the destination. If the channel is noisy, the physical layer will add some redundancy to its signals to reduce the bit error rate to a tolerable level.

However, the bit stream received by the data link layer is not guaranteed to be error free.

It is up to the data link layer to detect and, if necessary, correct errors.

Framing

The data link layer breaks up the bit stream into discrete frames, compute a short token called a checksum for each frame, and include the checksum in the frame when it is transmitted.

Breaking up the bit stream into frames is not straight forward.

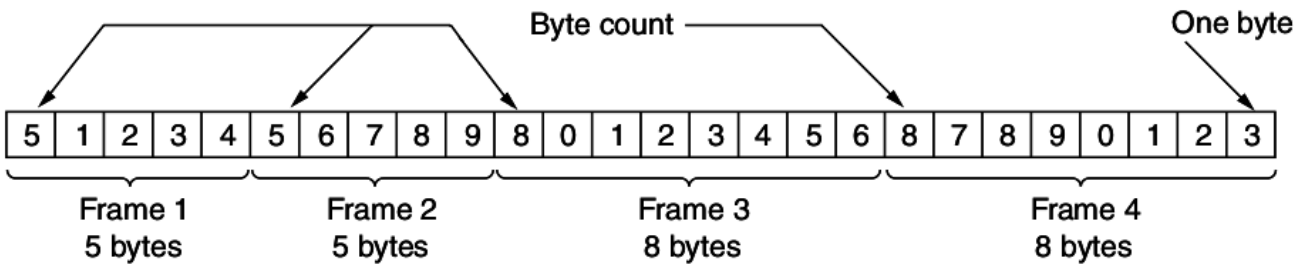
A good design must **make it easy for a receiver to find the start of new frames** while using little of the channel bandwidth.

Framing: 4 Methods

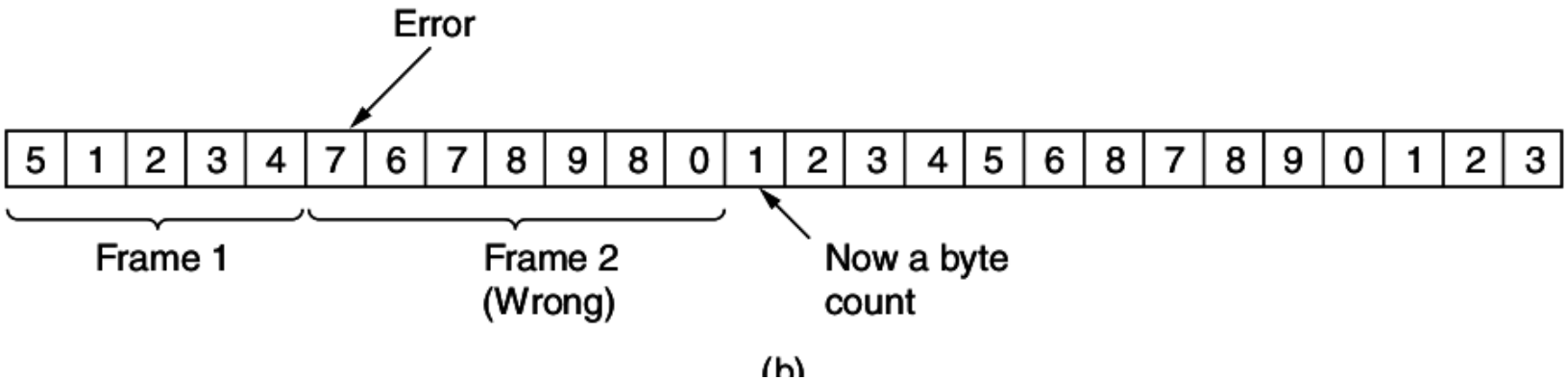
1. Byte count.
2. Flag bytes with byte stuffing.
3. Flag bits with bit stuffing.
4. Physical layer coding violations.

Byte count

- This framing method uses a field in the header to specify the number of bytes in the frame.
- When the data link layer at the destination sees the byte count, it knows how many bytes follow and hence where the end of the frame is.



Byte count

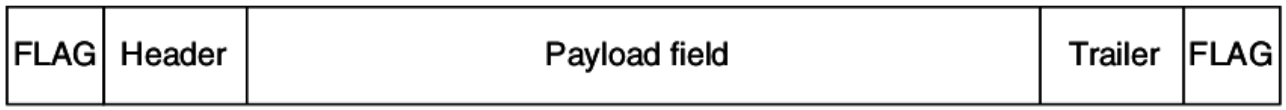

Error
5 1 2 3 4 7 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3
Frame 1
Frame 2
(Wrong)
Now a byte
count

(b)

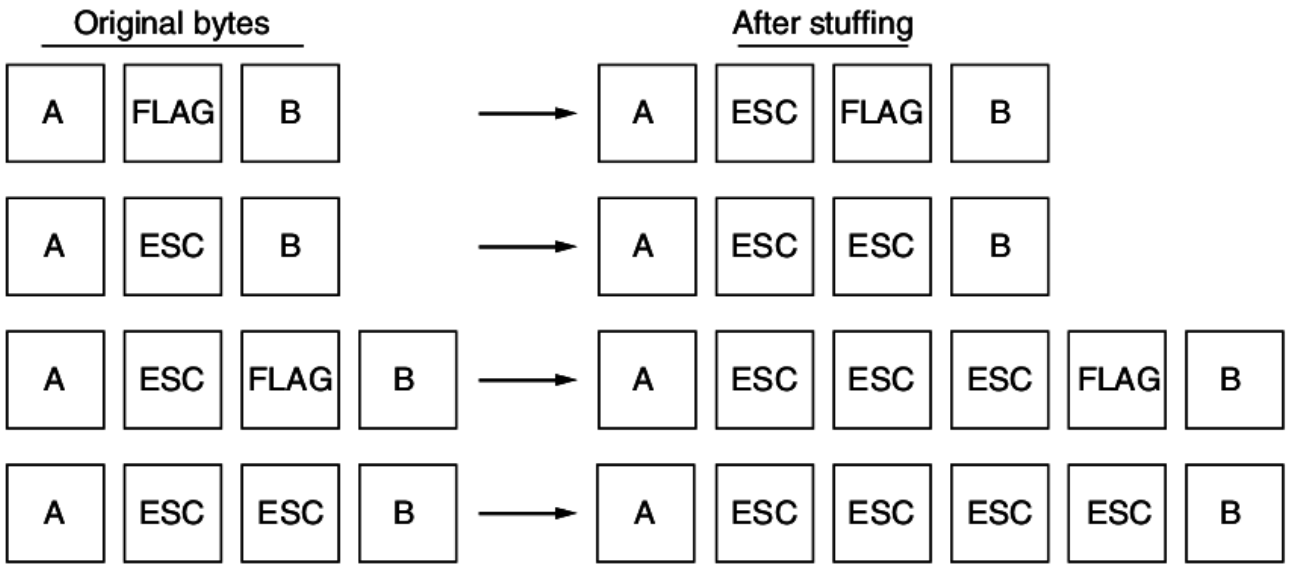
Flag bytes with byte stuffing

Resynchronization after an error is addressed by having each frame start and end with special bytes.

- **Flag byte**, is used as both the starting and ending delimiter.
- Two consecutive flag bytes indicate the end of one frame and the start of the next.
- If the receiver ever loses synchronization it can just search for two flag bytes to find the end of the current frame and the start of the next frame.

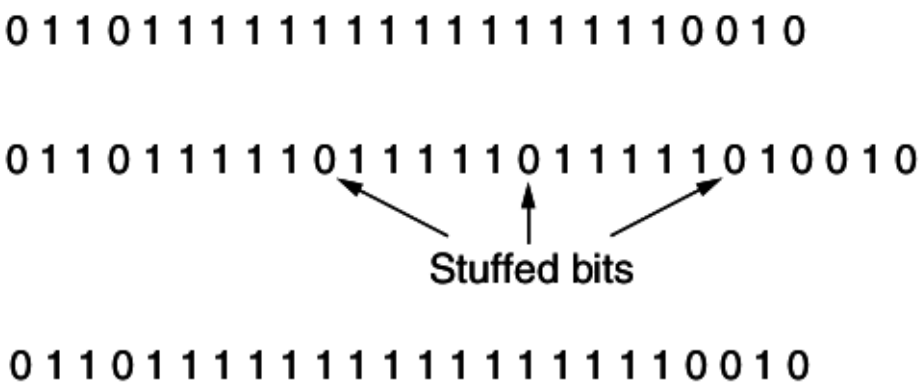


Flag bytes with byte stuffing



Flag bytes with bit stuffing

- Each frame begins and ends with a special bit pattern, **01111110 or 0x7E** in hexadecimal.
- 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.
- It also ensures a minimum density of transitions that help the physical layer maintain synchronization.



Physical layer coding violations

Redundancy in encoding of bits as signals are used to signal start and end of frames.

- In 4B/5B line code 4 data bits are mapped to 5 signal bits to ensure sufficient bit transitions.
- We can use some reserved signals to indicate the start and end of frames.
- It is easy to find the start and end of frames and there is no need to stuff the data.

Error Control

How to make sure all frames are eventually delivered to the network layer at the destination and in the proper order?

Error Control

- Positive or Negative acknowledgements.
- How to handle hardware errors or transmission losses?

Error Control

- Timers
- How to handle duplicate frames?

Error Control

- Sequence numbers.

Flow Control

- **Feedback based flow control:** the receiver sends back information to the sender giving it permission to send more data, or at least telling the sender how the receiver is doing.
- **Rate based flow control:** the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

Error Detection and Error Correction

Depends on channel characteristics what mechanism is employed.

- Two basic strategies for dealing with errors. Both add redundant information to the data that is sent.
- Error-correcting codes (Forward Error Correction) and Error-detecting codes .

Error Detection and Error Correction

- Neither correction nor detection can handle all possible errors since the redundant bits that offer protection are as likely to be received in error as the data bits.
- To avoid undetected errors the code must be **strong enough to handle the expected errors**.

Error types

- Extreme values of thermal noise that overwhelm the signal briefly and occasionally, giving rise to isolated single-bit errors.
- Errors that tend to come in bursts. Deep fade on a wireless channel or transient electrical interference on a wired channel.
- Erasure channel, analog signal that was far from the expected value for a 0 or 1 and declared the bit to be lost.

Error Detection and Error Correction

- Error-correcting codes are also seen in the physical layer, particularly for noisy channels, and in higher layers, particularly for real-time media and content distribution.
- Error-detecting codes are commonly used in link, network, and transport layers.

Error Correcting Codes

- Hamming codes.
- Reed-Solomon codes.
- Low-Density Parity Check codes.

All of these codes add redundancy to the information that is sent. A frame consists of **m** data (i.e., message) bits and **r** redundant (i.e. check) bits.

Block, Systematic, Linear Codes

- In a block code, the **r** check bits are computed solely as a function of the **m** data bits with which they are associated.
- In a systematic code, the **m** data bits are sent directly, along with the check bits, rather than being encoded themselves before they are sent.
- In a linear code, the **r** check bits are computed as a linear function of the **m** data bits.

Hamming Codes: Some Basics

- Let the total length of a block be n (i.e., $n = m + r$).
- An n-bit unit containing data and check bits is referred to as an n-bit codeword.
- The code rate, is the fraction of the codeword that carries information that is not redundant, or m/n.

Hamming Codes: Some Basics

- Given any two codewords that may be transmitted or received-say, 10001001 and 10110001, it is possible to determine how many corresponding bits differ.
- The number of bit positions in which two codewords differ is called the Hamming distance
- Its significance is that if two codewords are a Hamming distance **d** apart, it will require **d** single-bit errors to convert one into the other.

Hamming Codes

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.**

Hamming Codes

All 2^m possible data messages are legal, but due to the way the check bits are computed, not all of the 2^n possible codewords are used.

In fact, when there are r check bits, only the small fraction of$2^m/2^n$ or $1/2^r$ of the possible messages will be legal codewords.

It is the sparseness with which the message is embedded in the space of codewords that allows the receiver to detect and correct errors.

Importance of Hamming Distance

The error-detecting and error-correcting properties of a block code depend on its Hamming distance.

To reliably detect **d** errors, you need a distance **d+1** code because with such a code there is no way that **d** single-bit errors can change a valid codeword into another valid codeword. When the receiver sees an illegal codeword, it can tell that a transmission error has occurred.

Similarly, to correct d errors, you need a distance 2d + 1 code because that way the legal codewords are so far apart that even with d changes the original codeword is still closer than any other codeword.

Example

Consider a code with only four valid codewords: 0000000000, 0000011111, 1111100000, and 1111111111

Example

This code has a distance of 5, which means that it can correct double errors or detect quadruple errors.

If the codeword 0000000111 arrives and we expect only single or double-bit errors, the receiver will know that the original must have been 0000011111.

If, however, a triple error changes 0000000000 into 0000000111, the error will not be corrected properly.

Alternatively, if we expect all of these errors, we can detect them.

Code Design and How many Check Bits?

Imagine 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 $\mathbf{(n+1)}2^m \leq 2^n$. Using n=m+r, this requirement becomes $\mathbf{(m+r+1)} \leq 2^r$

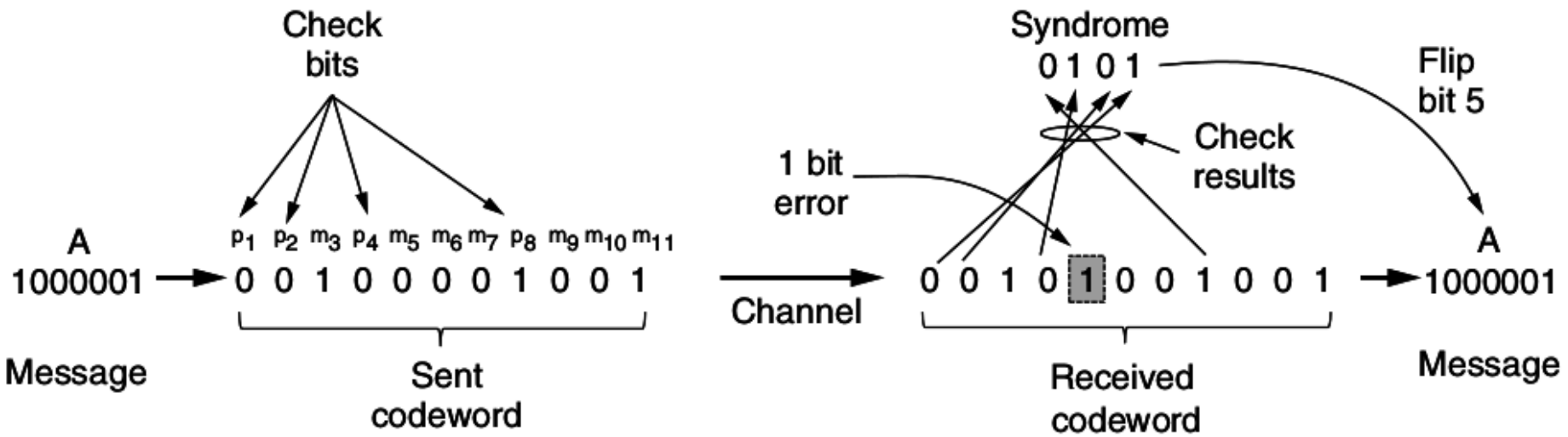
Given m, this puts a lower limit on the number of check bits needed to correct single errors.

Hamming Codes

In Hamming codes the bits of the codeword are numbered consecutively.

The bits that are powers of 2 (1, 2, 4, 8, 16, etc.) are check bits.

The rest (3,5,6,7,9, etc.) are filled up with the **m** data bits.

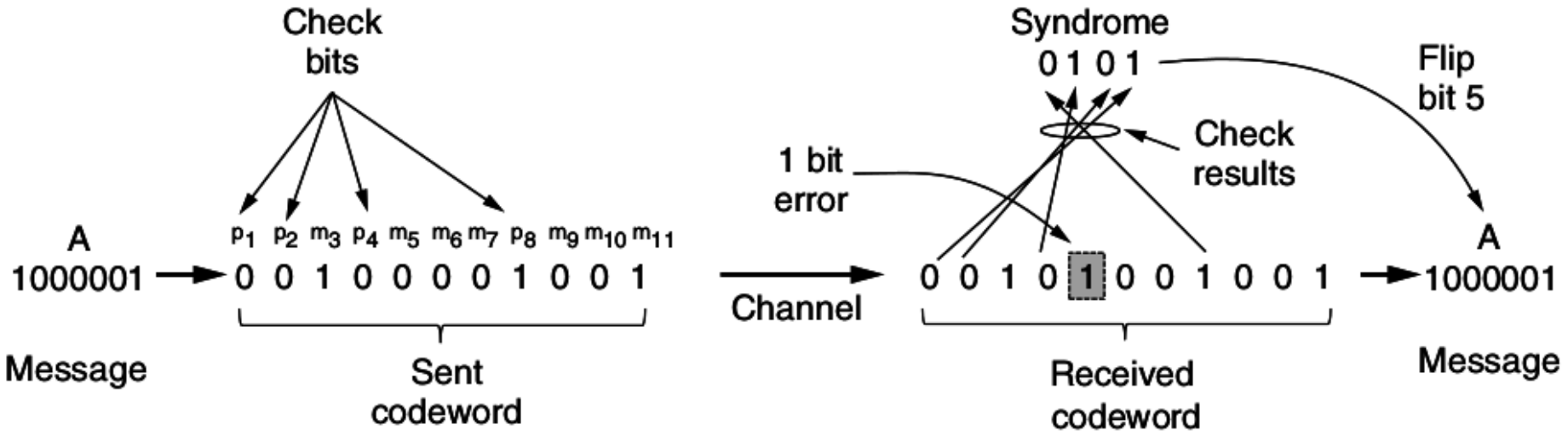


Hamming Codes

Each check bit forces the modulo 2 sum, or parity, of some collection of bits, including itself, to be even (or odd).

A bit may be included in several check bit computations.

To see which check bits the data bit in position k contributes to, rewrite k as a sum of powers of 2. For example, 11 = 1+2+8 and 29= 1+4+8+16. A bit is checked by just those check bits occurring in its expansion



Reed Solomon Codes

Reed-Solomon codes are also block-based error correcting codes with a wide range of applications in digital communications and storage. RS codes are used to correct errors in many systems including:

- Storage devices(including tape, Compact Disk, DVD, barcodes, etc)
- Wireless or mobile communications
- Satellite communications
- Digital television / DVB
- High-speed modems such as ADSL etc.

Error-Detecting Codes

3 different error-detecting codes, Parity, Checksums, Cyclic Redundancy Checks (CRCs).

Error-Detecting Codes

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).
- For example, when 1011010 is sent in even parity, a bit is added to the end to make it 10110100.
- A code with a single parity bit has a distance of 2, since any single-bit error produces a codeword with the wrong parity. This means that it can detect single-bit errors.

Error-Detecting Codes

Consider a channel on which errors are isolated and the error rate is 10^{-6} per bit (Typical LAN links provide bit error rates of 10^{-10}).

- Let the block size be 1000 bits.
- To provide **error correction** for **1000-bit blocks**, what is number of check bits required?
- How many check bits are required for megabit of data?
- To merely detect a block with a single 1-bit error, one parity bit per block will suffice.
- Once every 1000 blocks, a block will be found to be in error and an extra block (1001 bits) will have to be transmitted to repair the error.

Error-Detecting Codes

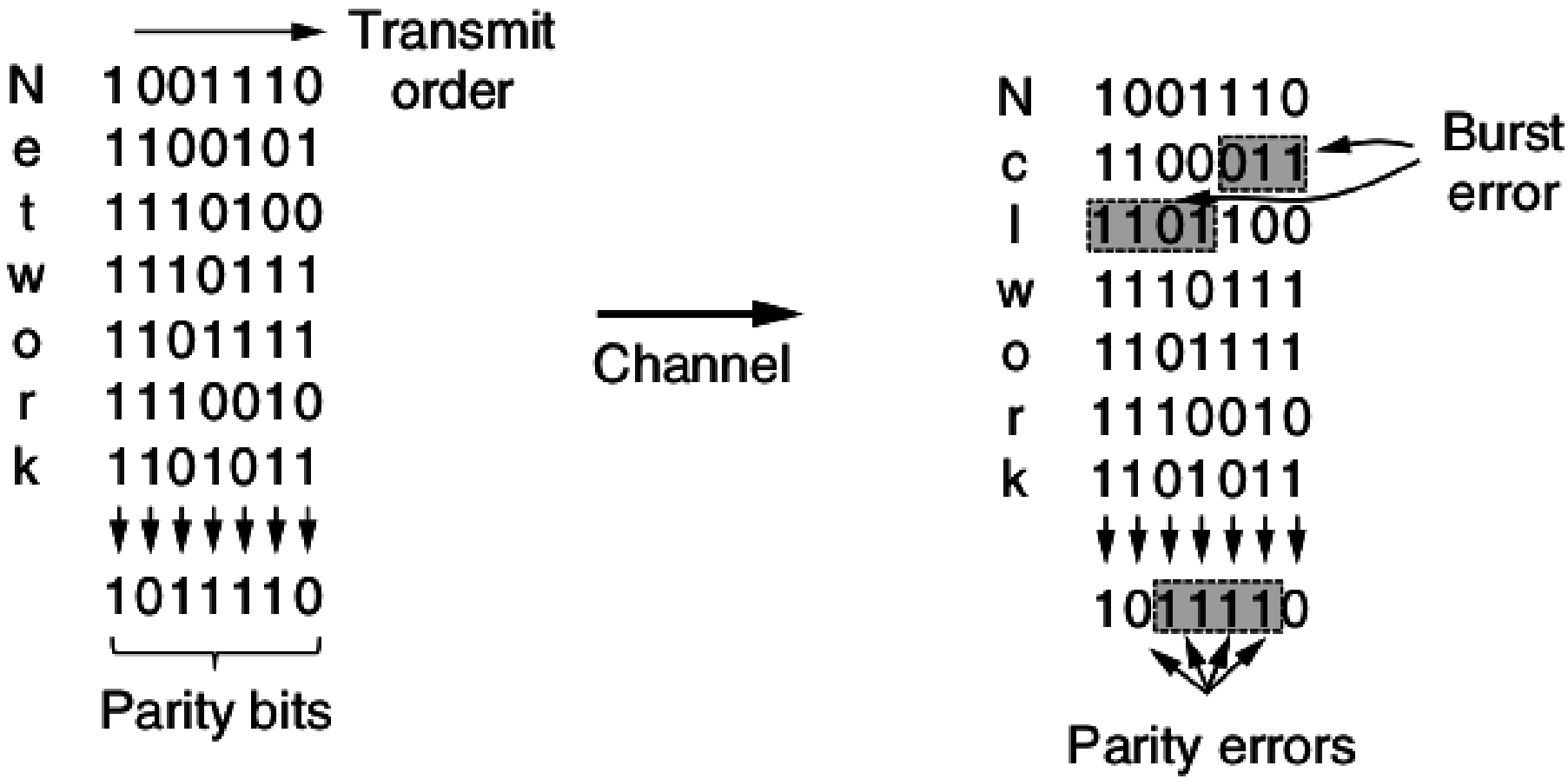
- A single parity bit can only reliably detect a single-bit error in the block.
- If the block is badly garbled by a long burst error, the probability that the error will be detected is only 0.5, which is hardly acceptable.

Error-Detecting Codes:Burst Errors

To better protect against burst errors, we can compute the parity bits over the data in a different order than the order in which the data bits are transmitted.

- **Interleaving** will compute a parity bit for each of the **n** columns and send all the data bits as **k** rows, sending the rows from top to bottom and the bits in each row from left to right in the usual manner.
- At the last row, we send the **n** parity bits.
- Interleaving is a general technique to convert a code that detects (or corrects) isolated errors into a code that detects (or corrects) burst errors.

Error-Detecting Codes



Error-Detecting Codes:Checksum

Checksum is a group of check bits associated with a message, regardless of how those bits are calculated.

- **Ex:** A group of parity bits
- **Ex:** Running sum of the data bits of the message. The checksum is usually placed at the end of the message, as the complement of the sum function.This way, errors may be detected by summing the entire received codeword, both data bits and checksum. If the result comes out to be zero, no error has been detected.

IP Checksum

IP header checksum is calculated over IP header only.

- IP header checksum is: 16 bit one's complement of the one's complement sum of all 16 bit words in the header.
- The result of summing the entire IP header, including checksum, should be zero if there is no corruption.

For example:
4500 0073 0000 4000 4011 **b861** c0a8 0001 c0a8 00c7

To calculate the checksum, 4500 + 0073 + 0000 + 4000 + 4011 + c0a8 + 0001 + c0a8 + 00c7 = 2479C = 2 + 479C = 479E.

One's complement of this result: B861.

Error-Detecting Codes

CRC (Cyclic Redundancy Check), also known as a 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.
- Such a polynomial is said to be of degree k-1.
- The high-order (leftmost) bit is the coefficient of x^{k-1}, the next bit is the coefficient of x^{k-2}, and so on.
- Write polynomial for 110001?

Error-Detecting Codes

Polynomial arithmetic is done modulo 2. It does not have carries for addition or borrows for subtraction. Both addition and subtraction are identical to exclusive OR.

- Long division is carried out in exactly the same way as it is in binary except that the subtraction is again done modulo 2.
- When the polynomial code method is employed, the sender and receiver must agree upon a generator polynomial, G(x), in advance.
- Both the high and low-order bits of the generator must be 1.
- To compute the CRC for some frame with m bits corresponding to the polynomial M(x), the frame must be longer than the generator polynomial.

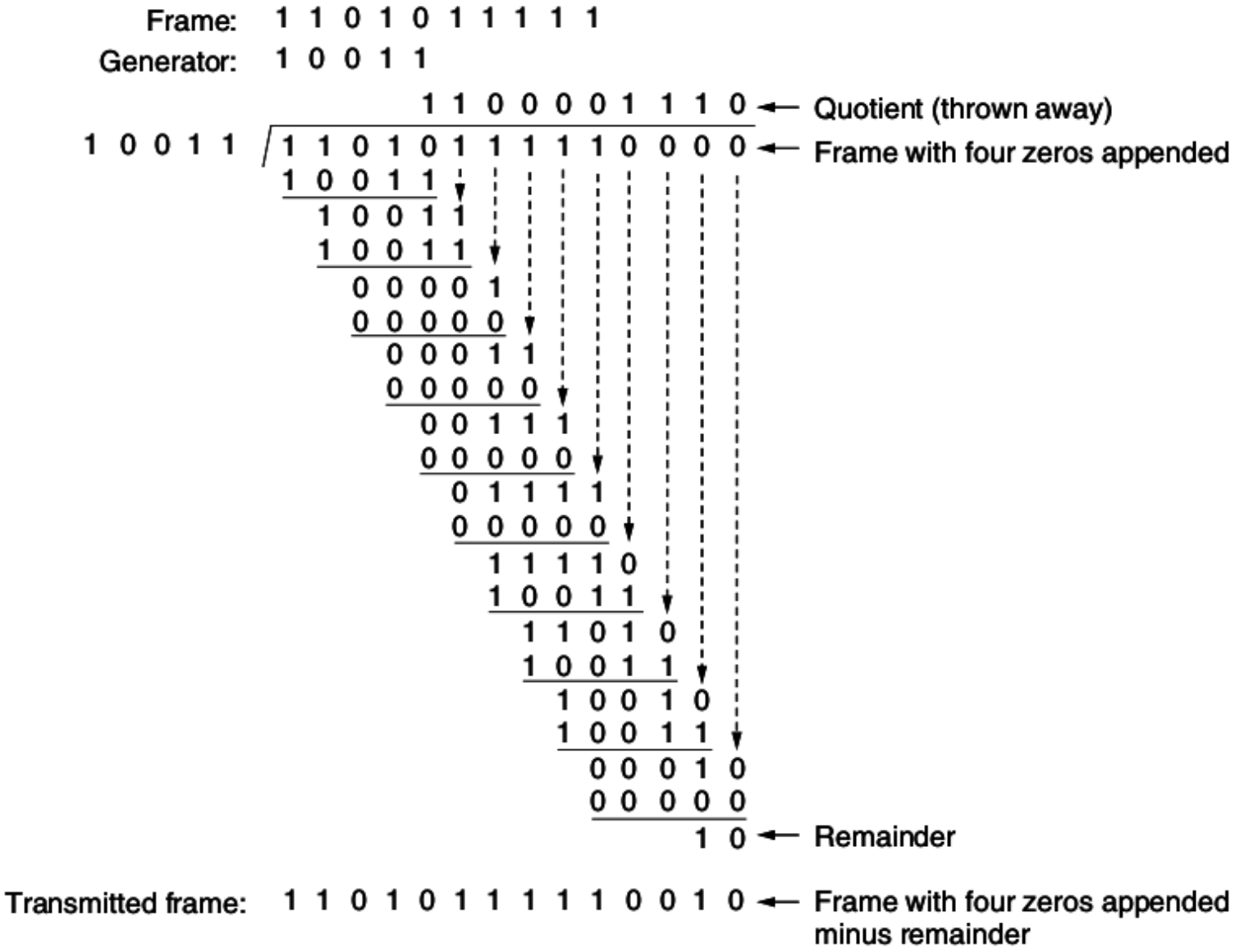
Error-Detecting Codes

- 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 the checksummed frame, it tries dividing it by G(x). If there is a remainder, there has been a transmission error.

Algorithm for computing CRC

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 $x^r M(x)$.
2. Divide the bit string corresponding to $G(x)$ into the bit string corresponding to $x^r M(x)$, using modulo 2 division.
3. Subtract the remainder (which is always r or fewer bits) from the bit string corresponding to $x^r M(x)$ using modulo 2 subtraction. The result is the checksummed frame to be transmitted. Call its polynomial $T(x)$.

Algorithm for computing CRC



The End