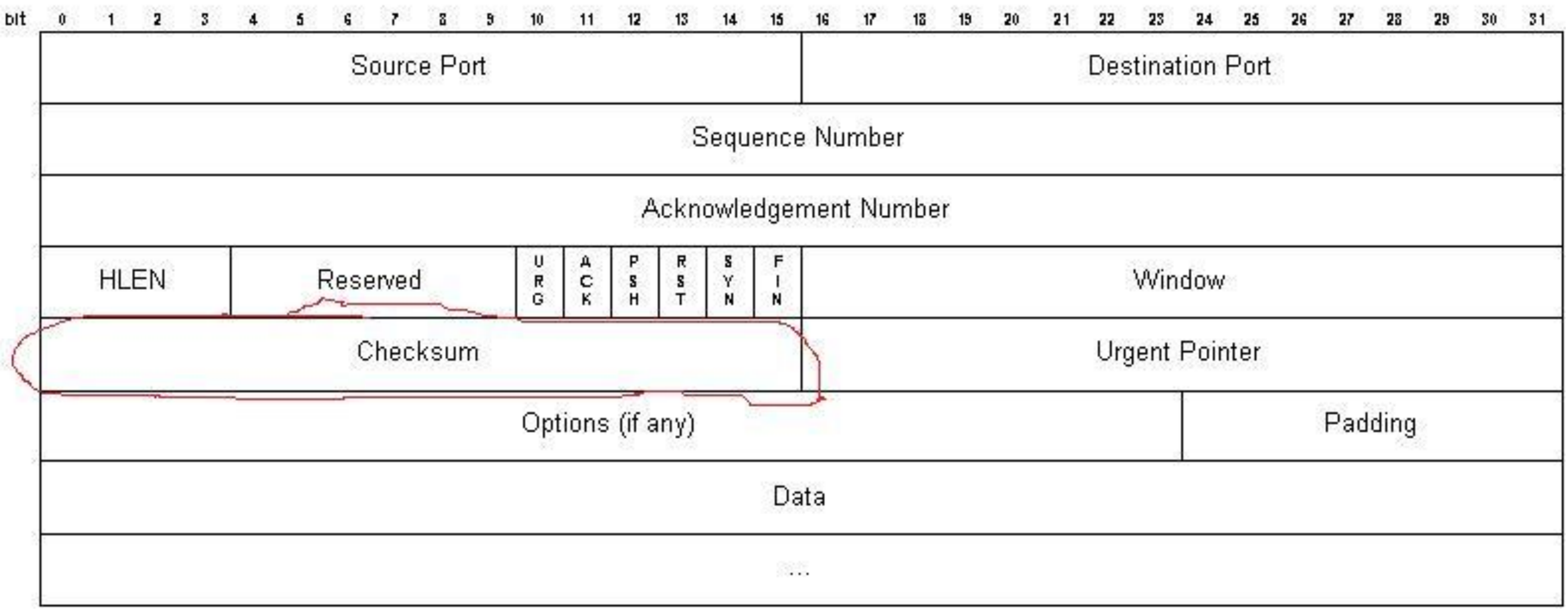
Error-correcting codes introduction, Hamming distance

Mark Sevalnev

05.03.2008

TCP’s headers

source: http://condor.depaul.edu/~jkristof/technotes/tcp-segment-format.jpg

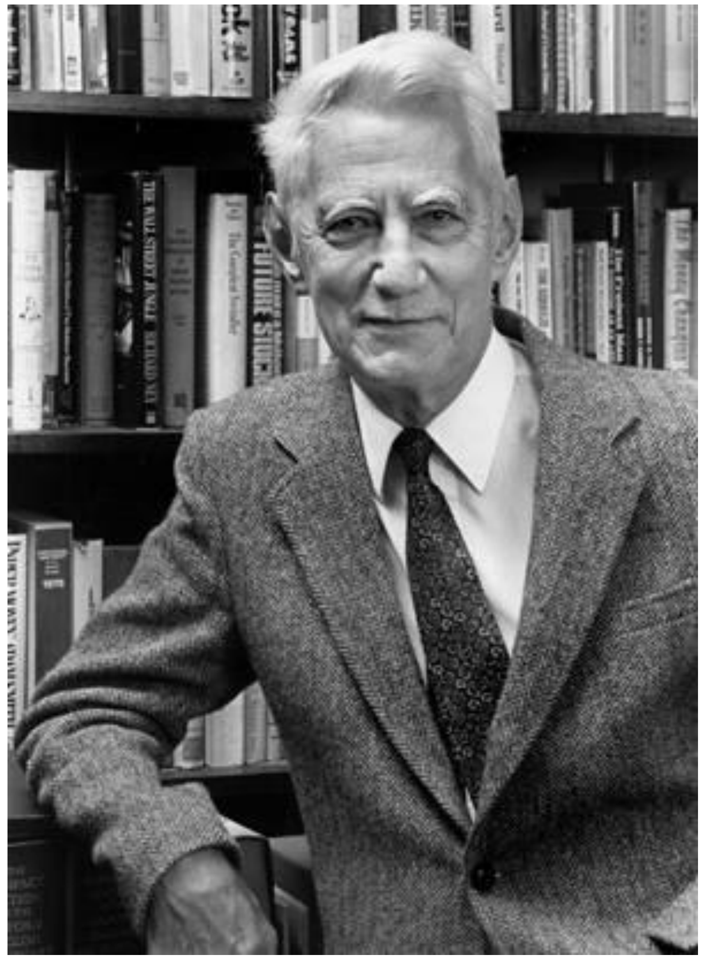


Assumptions:

- No eavesdropper
- Methods for transmitting data are susceptible to outside influences that can cause errors
- Error-correcting codes: mathematical method <u>of detecting errors</u> and <u>correcting errors</u>
- Error-correcting codes began with Claude Shannons's famous paper: ”A Mathematical Theory of Communication” (1948)

Claude Shannon

source: http://en.wikipedia.org/wiki/Claude_Shannon



Example 1:

- A -> 000
- B -> 001
- C -> 010
- D -> 011
- E -> 100
- F -> 101
- G -> 110
- H -> 111

- The transmission is binary
- Every letter is encoded in a string of the same length
- The sender and receiver agrees on this way to code the eight letters
- If no errors occur, this is a perfectly good way to code the eight letters

Example 1 (cont.):

Sender wants to send the message ‘bad’,
so sending the string 001 000 011.

If no errors occur, then receiver gets the string
001 000 011, then he breaks it up into blocks of three: 001, 000 and 011. He knows 001 represents B, 000 represents A, and 011 represents D and so he decodes the message correctly.

Example 1(cont.):

But what happens if an error occurs?

For instance sending 001 000 011

but receiving 101 000 011,

and after decoding… 101 -> F, 000 -> A, 011 -> D.

Not only does the sender get the wrong message, but he is not even aware that an error has occurred.

Example 2:

- A -> 000 000
- B -> 001 001
- C -> 010 010
- D -> 011 011
- E -> 100 100
- F -> 101 101
- G -> 110 110
- H -> 111 111

- Doubling the size of the letter’s representation
- Can detect the error but inefficient
- Detects one error
- Can’t to correct an error

Example 2 (cont.):

Sender wants to send the message ‘bad’,

so sending the string 001 001 000 000 011 011.

One error occurs, receiver gets the string

101 001 000 000 011 011.

Receiver breaks up the message into blocks of six: 101 001, 000 000 and 011 011. Now, the receiver knows that 000 000 is A and 011 011 is D. But the string 101 001 was not assigned to any letter, so an error must occurred

Example 3:

- A -> 0000
- B -> 0011
- C -> 0101
- D -> 0110
- E -> 1001
- F -> 1010
- G -> 1100
- H -> 1111

- The binary parity check: there are an even number of ones in each string of four digits
- Can detect a single error
- Uses strings of length 4 instead of 6

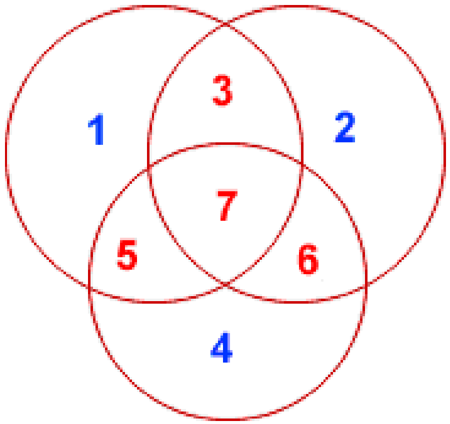
Example 4:

- A -> 000 000 000
- B -> 001 001 001
- C -> 010 010 010
- D -> 011 011 011
- E -> 100 100 100
- F -> 101 101 101
- G -> 110 110 110
- H -> 111 111 111

- The way to not only detect an error but also correct it
- Uses blocks of length 9 to represent each letter
- Can correct any single error

Example 5:

- 0 0 0 0 0 0 0 **1**
- 0 0 0 0 1 1 1 **2**
- 0 0 1 1 0 0 1 **3**
- 0 0 1 1 1 1 0 **4**
- 0 1 0 1 0 1 0 **5**
- 0 1 0 1 1 0 1 **6**
- 0 1 1 0 0 1 1 **7**
- 0 1 1 0 1 0 0 **8**
- 1 0 0 1 0 1 1 **9**
- 1 0 0 1 1 0 0 **A**
- 1 0 1 0 0 1 0 **B**
- 1 0 1 0 1 0 1 **C**
- 1 1 0 0 0 0 1 **D**
- 1 1 0 0 1 1 0 **E**
- 1 1 1 1 0 0 0 **F**
- 1 1 1 1 1 1 1 G



•Binary [7, 4] Hamming code

Source: http://www.ece.unb.ca/tervo/ee4253/hamming.htm

THE PURPOSE:

OUR GOAL IS TO FIND A WAY TO TRANSMIT INFORMATION IN A REASONABLY EFFICIENT WAY SO THAT WE CAN ALSO CORRECT A REASONABLE NUMBER OF ERRORS

- **definition**: Let A be any set and n >= 1 an integer. We define

  A^n = { (a1, a2, … , an) | ai ∈ A }

- **definition**: A is a finite set. n >= 1 is an integer.

  A *code* C of length n is any subset of A^n. A^n is the *codespace* and the elements of A^n are *words*. Elements of *code* C are *codewords.*

Example 1 (revised):

- A -> 000
- B -> 001
- C -> 010
- D -> 011
- E -> 100
- F -> 101
- G -> 110
- H -> 111

- A = {0, 1}
- n = 3
- A^n = {000, 001, 010, 011,
  100, 101, 110, 111}
- C = {000, 001, 010, 011,
  100, 101, 110, 111}
- C = A^n
- All possible *words* received are *codewords* => we cannot tell if an error occur

Example 3(revised):

- A -> 0000
- B -> 0011
- C -> 0101
- D -> 0110
- E -> 1001
- F -> 1010
- G -> 1100
- H -> 1111

- A = {0, 1}
- n = 4
- A^n = {0000, 0001, 0010, 0011, 0100 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111}
- C = {0000, 0011, 0101, 0110,
  1001, 1010, 1100, 1111}
- C <> A^n

Hamming distance

In information theory, the *Hamming distance* between two strings of equal length is the number of positions for which the corresponding symbols are different. Put another way, it measures the minimum number of *substitutions* required to change one into the other, or the number of *errors* that transformed one string into the other.

Source: http://en.wikipedia.org/wiki/Hamming_distance

Hamming distance (formal definition):

- **definition:** Let x, y $\in$ A^n. We define the *Hamming distance* between x and y, denoted dH(x, y), to be the number of places where x and y are different.

Few examples:

The *Hamming distance* between:

- **1011101** and **1001001** is ...
- **2173896** and **2233796** is ...
- "**toned**" and "**roses**" is ...

Source: http://en.wikipedia.org/wiki/Hamming_distance

Answers:

The *Hamming distance* between:

- **1011101** and **1001001** is 2
- **2173896** and **2233796** is 3.
- "**toned**" and "**roses**" is 3.

Source: http://en.wikipedia.org/wiki/Hamming_distance

Metric

- A **metric** on a set X is a function (called the *distance function* or simply **distance**)

  $d : X \times X \to \mathbf{R}$

  (where **R** is the set of real numbers). For all x, y, z in X, this function is required to satisfy the following conditions:
- $d(x, y) \geq 0$ (*non-negativity*)
- $d(x, y) = 0$ if and only if $x = y$ (*identity of indiscernibles*. Note that condition 1 and 2 together produce *positive definiteness*)
- $d(x, y) = d(y, x)$ (*symmetry*)
- $d(x, z) \leq d(x, y) + d(y, z)$ (*subadditivity* / *triangle inequality*).

Source: http://en.wikipedia.org/wiki/Metric_(mathematics)

CONSEQUENCE: HAMMING DISTANCE IS A METRIC

- Using the concept of *Hamming distance*, we can mathematically describe the method we will use for error correction.
- When a *word* r is received, we decode it by finding a *codeword* x such that dH(r, x) is the smallest possible.
- This method is called *minimum distance decoding*.
- Notice that given a received *word* r, there may be more than one valid *codeword* whose *Hamming distance* to r is the smallest possible => cannot correct the *word* with confidence.

- **definition:** Let C be a *code* and a subset of A^n. We define the *minimum distance* of the code to be:

  min {dH(x, y)}, x, y ∈ C, x <> y

- **theorem:** Let C be a *code* and subset of A^n, and with *minimum distance* d. Then C detects d – 1 errors

- **theorem:** Let C be a *code* and subset of A^n, and with *minimum distance* d. Then C corrects e errors where s = floor{(d - 1) / 2}

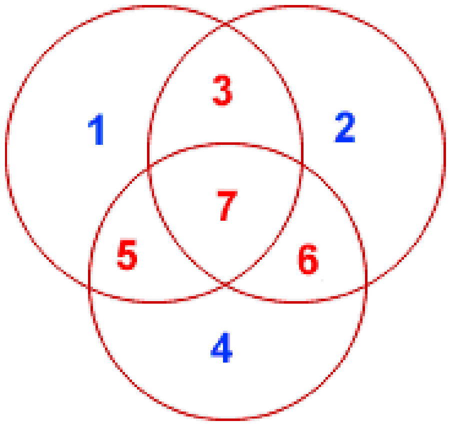
Example 3(revised):

- A -> 0000
- B -> 0011
- C -> 0101
- D -> 0110
- E -> 1001
- F -> 1010
- G -> 1100
- H -> 1111

- Minimum distance 2
- Detects a single error
- Will not correct any error

Example 5(revised):

- 0 0 0 0 0 0 0 **1**
- 0 0 0 0 1 1 1 **2**
- 0 0 1 1 0 0 1 **3**
- 0 0 1 1 1 1 0 **4**
- 0 1 0 1 0 1 0 **5**
- 0 1 0 1 1 0 1 **6**
- 0 1 1 0 0 1 1 **7**
- 0 1 1 0 1 0 0 **8**
- 1 0 0 1 0 1 1 **9**
- 1 0 0 1 1 0 0 **A**
- 1 0 1 0 0 1 0 **B**
- 1 0 1 0 1 0 1 **C**
- 1 1 0 0 0 0 1 **D**
- 1 1 0 0 1 1 0 **E**
- 1 1 1 1 0 0 0 **F**
- 1 1 1 1 1 1 1 G



•Binary [7, 4] Hamming code
•Minimum distance 3
•Detects 2 errors
•Corrects a single error

Source: http://www.ece.unb.ca/tervo/ee4253/hamming.htm